AX-2000/AX-2000 plus Computer Interface Experiment set

CD-ROM information

Sourecode and Experiment program information in INEXÌs Computer Interface CD-ROM

In CD-ROM is contained all sourcecode and demo software that compile and run with Visual BASIC V6.0 or higher (not test with VB.net). The sourcecodes are contained in

Parallel_port/Lab for Parallel port interface experiment

Serial_port/Lab for Serial port interface experiment

USB_port/Lab for USB port interface experiment

For demo execute files from sourcecode are contained in

Parallel_port/Software/Demo for Parallel port interface experiment

Serial_port/Software/Demo for Serial port interface experiment

1. Software tools, drivers and control files

`io.dll` : Input/Output DLL for Parallel port (**Parallel_port/Software/DLL)**

OCX file are contained in **Parallel_port/Software/OCX**.

`Microchip HIDComm ActiveX setupex.exe` : HIDComm setup file (**USB_port/Software/Microchip HIDComm ActiveX**)

`hpcount.ocx` is contained in **USB_port/Software/OCX**.

2. Parallel port experiment

Lab01 : Sending output data to Data port

Lab02 : Sending output data to Control port

Lab03 : Reading data via Status port

Lab04 : Driving relay

Lab05 : 1-Phase stepper motor driver

Lab06 : 2-Phase stepper motor driver

Lab07 : Half step mode stepper motor driver

Lab08 : Drive 1-digit LED 7 segment

Lab09 : Drive LED 7 segment in multiplexed

Lab10 : N/A

Lab11 : Port expansion via I^{2}C bus

Lab12 : Analog interface via I^{2}C bus

Lab13 : Output port expansion

Lab14 : Drive stepper motor with EX-09

Lab15 : Parallel port versus DS1621

3. Serial port experiment

Lab01 : Simple transmit serial data

Lab02 : Building PCís timer control

Lab03 : Reading logic to serial port

Lab04 : Receiving data with UTX8100 controller

Lab05 : Receiving data from UART IN

Lab06 : Simple stepper motor controller

Lab07 : Serial port drives relay

Lab08 : Receiving data and error verification

Lab09 : Port expansion via I^{2}C bus

Lab10 : Analog interface via I^{2}C bus

Lab11 : Output port expansion

Lab12 : Output port expansion with S-Board V2.0 special function

Lab13 : Serial port with temperature measurement

4. USB port experiment

Lab01 : Connect U-Board with HIDComm control

Lab02 : PORTOUT experiment - transmit data to output

Lab03 : PORTIN experiment - get data from input

Lab04 : PORTIN experiment with Report ID0

Lab05 : Simple USB-based stepper motor controller

Lab06 : Simple USB-based relay controller

Lbb07 : Analog interface with USB port

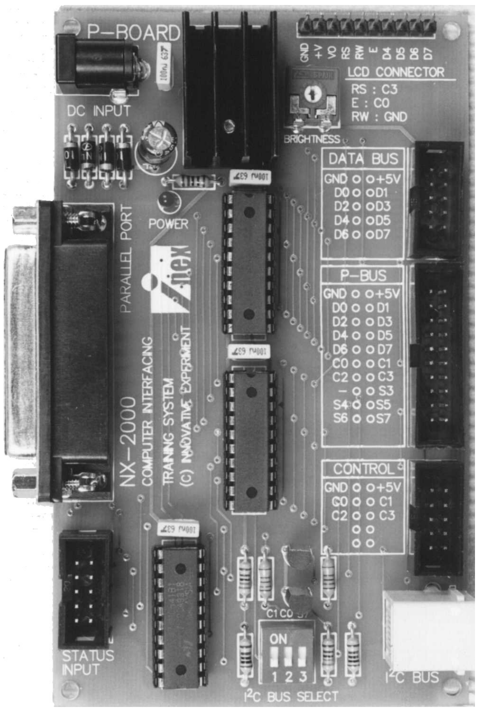
Lab08 : Output port expansion for USB

Addition, can download all sourcecode, demo software, driver and related info at www.inexglobal.com

P-board

Parallel port interface board

(C) Innovative Experiment Co.,Ltd.



Features :

- In-built Buffer Circuit for all pins on port
- On-board I^{2}C bus converter circuit
- Polarity protection circuit
- Extension connector for DATA BUS, P-BUS, CONTROL and STATUS For connecting EX-series board

Packing List

- P-board
- Documentation
- CX-25 custom parallel port cable x 1
- IDC-20 cable x 1
- IDC-10 cable x 2
- CD-ROM

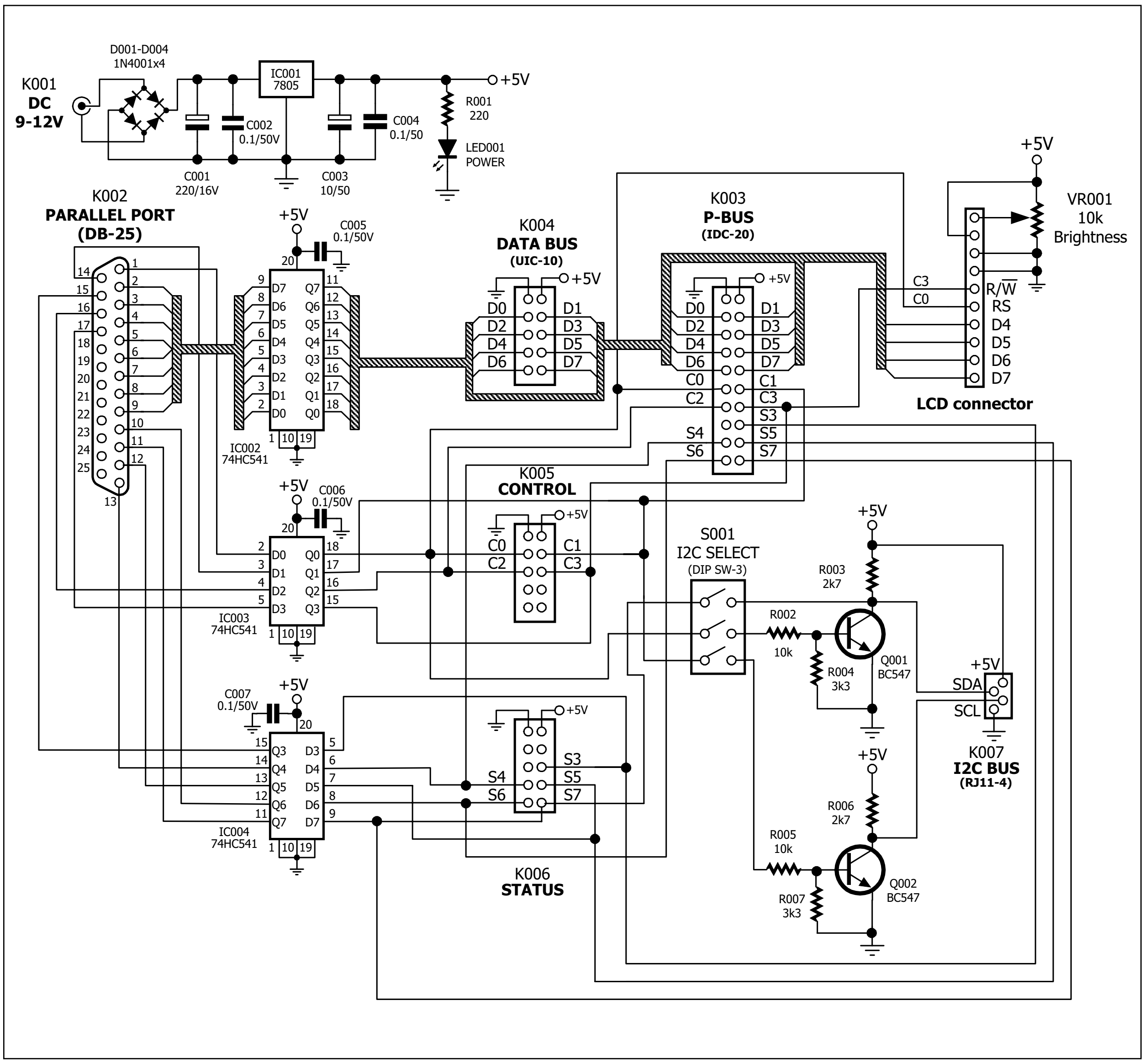


Figure-1 Schematic of P-Board Parallel port interface board

Board description

The P-Board will be interface with the computer parallel port directly and has a buffer circuit to prevent the parallel port signal from error. This board provides 3 ports to transfer signal of parallel port.

1. **Data port** has 8 signal pins, and is called D0-D7. It is output port only. All signal are connected directly to DATA BUS connector. Their pin are assigned in UIC-10 pin standard (It is INEXís custom standard pin) and combined the signal into P-BUS. It contains all signal of parallel port pin.

2. **Control port has 4** signal pins, and is called C0 to C3. C0, C1 and C3 is an invert logic pin. It is an output port only same Data port. Their pin are assigned in UIC-10 pin standard but use only 4 pins and combined the signal into P-BUS. It contains all signal of parallel port pin.

Addition Control port pin are used to I2C bus signal pin. C1 is config to SCL (Serial clock), C0 is config to SDA (Serial data output) and S7 (from Status port) is config to Input SDA (Serial data input). In using I^2C bus signal, selected by 3-points DIP switch.

3. **Status port** has 5 signal pins, and is called S3 to S7. Their pin are assigned in UIC-10 pin standard but use only 5 pins and combined the signal into P-BUS. It contains all signal of parallel port pin.

Addition a Status port pin S7 is used to I2C bus signal pin. It is config to Input SDA (Serial data input). In using I^2C bus signal, selected by 3-points DIP switch.

P-Board need +9V to +12V 500mA external DC power supply. Apply the supply voltage via DC jack. P-board has +5V regulator circuit. +5V regulated supply voltage is used for all circuit and some extension board interfaced via P-BUS , DATA BUS and I^2C bus.

Step by step of simple parallel port interfacing

Normally in Visual BASIC programming, you do not prepare an OUT instruction. Add INPOUT32.BAS file into the current Visual BASIC project. The step is :

(1) Copy *io.dll* file from the bundled CD-ROM into **the SYSTEM** directory in Windows.

(2) Enter **Project menu**, select **Add File** for add *INPOUT32.BAS* file into the Project. At **Project window** will appear *INPOUT32.BAS* file. See the figure-2

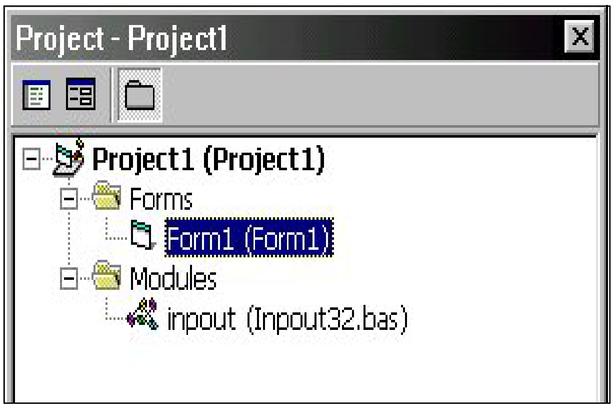
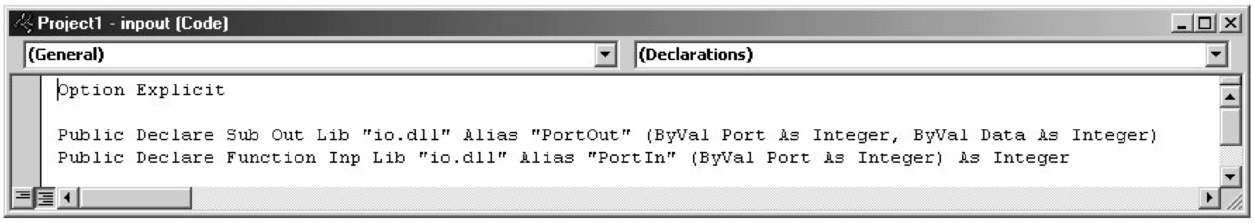


Figure-2 INPOUT.BAS launching

(3) Point to *INPOUT32.BAS* file. Use **View Code command** to see the detail of *INPOUT32.BAS* file. See the figure-3

(4) *INPOUT32.BAS* file wil prepare *INP* and *OUT* instruction for Visual BASIC. However please ensure you ahve copied the io.dll file into the SYSTEM directory.



```
Option Explicit

Public Declare Sub Out Lib "io.dll" Alias "PortOut" (ByVal Port As Integer, ByVal Data As Integer)
Public Declare Function Inp Lib "io.dll" Alias "PortIn" (ByVal Port As Integer) As Integer
```

Figure-3 Show detail of INPOUT.BAS file

Sample experiment-1

Data port sending value with OUT instruction of Visual BASIC via P-Board

Material :

1. P-Board Parallel port Interface board x 1
2. EX-01 board x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher
4. IDC-10 cable x 1

Procedure

1.1 Connect P-board with PCís parallel port and connect P-board with EX-01 by IDC-10 cable at DATA BUS connector. See figure-2

1.2 Open Visual BASIC.

1.3 Enter **Project** menu. Select **Add File** to add *INPOUT32.BAS* file (contain in bundle CD-ROM of P-board) into **Project**. At **Project window** will appear *INPOUT32.BAS* file.

1.4 Place 2 Command Buttons into Form1 following figure-3.

1.5 Double click at **`Command1`** to enter **View Code menu**. Write the code below for **`Command1_Click`** event

```
Private Sub Command1_Click()
Out &H378, &HFF
End Sub
```

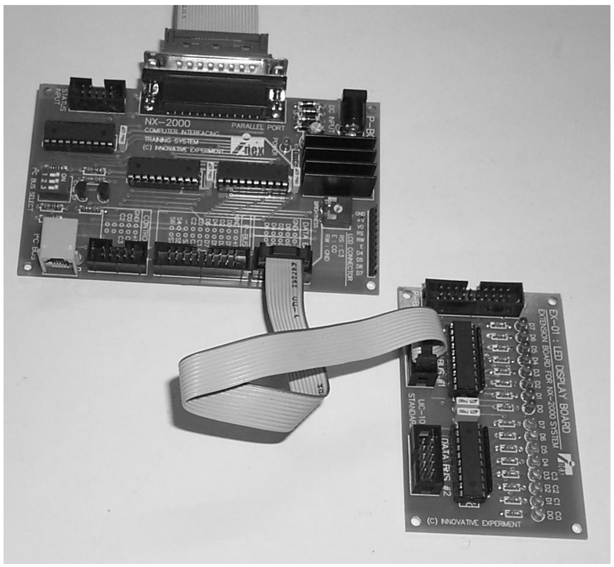


Figure-2 P-board interface with EX-01 board.

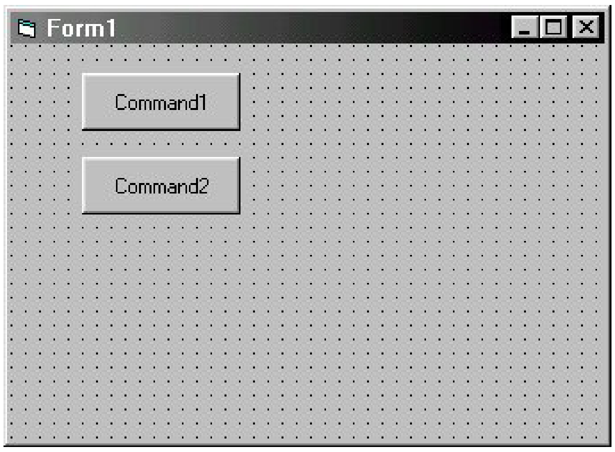
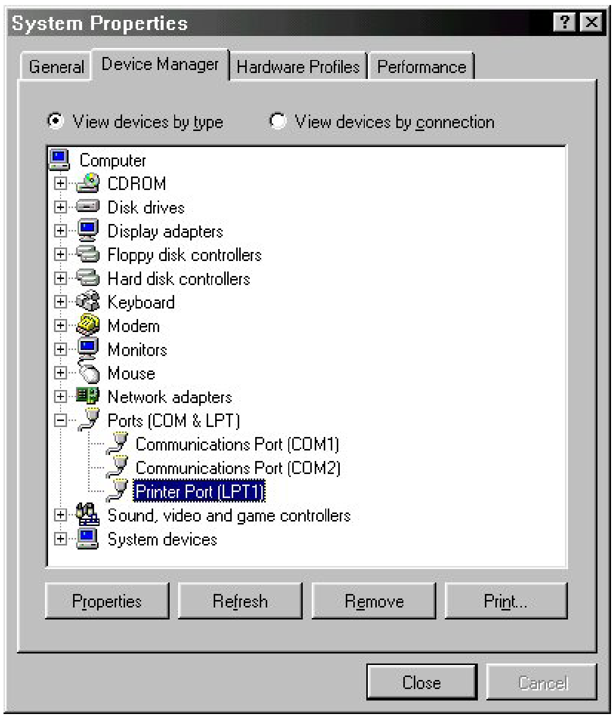


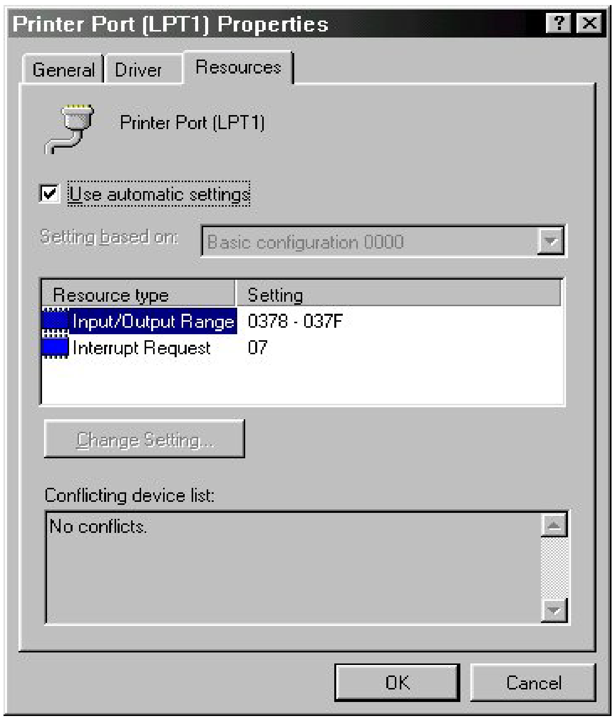
Figure-3 Placing Command Button in Visual BASIC


System Properties

General | Device Manager | Hardware Profiles | Performance

(●) View devices by type ( ) View devices by connection

```
Computer
+ CDROM
+ Disk drives
+ Display adapters
+ Floppy disk controllers
+ Hard disk controllers
+ Keyboard
+ Modem
+ Monitors
+ Mouse
+ Network adapters
- Ports (COM & LPT)
    Communications Port (COM1)
    Communications Port (COM2)
    Printer Port (LPT1)
+ Sound, video and game controllers
+ System devices
```


Printer Port (LPT1) Properties

General | Driver | Resources

Printer Port (LPT1)

[✓] Use automatic settings

Setting based on: Basic configuration 0000

Resource type	Setting
Input/Output Range	0378 - 037F
Interrupt Request	07

Conflicting device list:

No conflicts.

Figure-4 Device Manager window for checking the hardware

Figure-5 Resource window of Parallel port for report the parallel port

6. Double click at **Command2** and write the sourcecode for **Command2_Click** event below :

```
Private Sub Command2_Click()
Out &H378, 0
End Sub
```

7. Apply voltage for P-Board

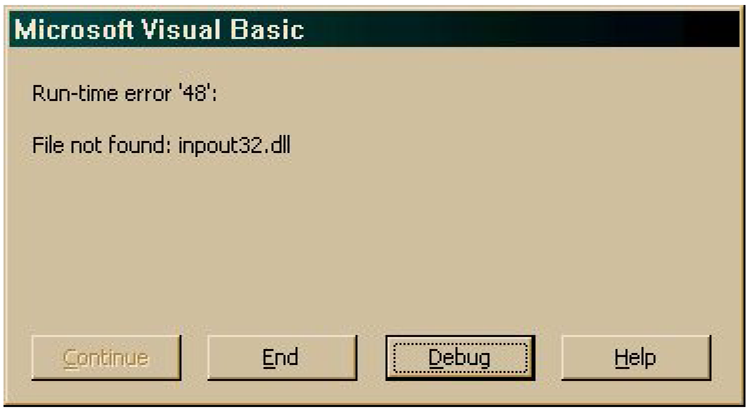
8. Run the program. Click **Command1** button. Observe the result at LED on EX-01 board.

8-LEDs on EX-01 board turn on.

9. Click **Command2** button. Observe the result at LED on EX-01 board.

8-LEDs on EX-01 board turn off

10. If program done not correct, check the parallel port address normally is 378H. Click My Computer and right click to select Property and select to Device Manager. The window in figure-4 will appear. Select Printer Port and click Property button and select to Resource tab. The window in figure-5 will appear. In this window will show the parallel port address the using. First number is DATA port address (0378). If user use different must change.


Microsoft Visual Basic

Run-time error '48':

File not found: inpout32.dll

Figure-6 Warning message of Inpout32.dll file not founding.

11. If the warning dialogue box in figure-6 appeares, it means that the user has not copied the Inpout32.dll fiile into SYSTEM folder of Windows.

12. If all is OK, change the value to send to EX-01 such as `OUT &H378,&H55.`

The LED will turn on and off swap. *Because 55H value is 01010101 in binary. "1" means LED on and "0" means LED off.*

Sample experiment-2

Control port sending value with OUT instruction of Visual BASIC via P-Board

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-01 board	x 1

3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

4. IDC-10 cable	x 1

Procedure

2.1 Connect P-board with PCís parallel port and connect P-board with EX-01 by IDC-10 cable at DATA BUS connector. See figure-2

2.2 User can use program from experment-1. Edit **`Command1_Click`** to

```
Private Sub Command1_Click()
Out &H37A, &HF4
End Sub
```

2.3 Edit **`Command2_Click`** to :

```
Private Sub Command1_Click()
Out &H37A, &HB
End Sub
```

2.4 Run the program. Try to click the **`Command1`** and **`Command2`** button. Observe the operation.

<u>Command2 button is pressed</u>

All LED on EX-01 board turn-off. Because C0, C1 and C3 is an invert bit.

<u>Command1 button is pressed.</u>

LED on EX-01 board will turn-on only 4 positions. This is because Control port of parallel port has only 4-bits.

Sending data of Control port is similar Data port. The different is the Control port address next Data port 2 address. If Data port address is at 378H, Control port address will be 37AH. Another different is number of pins. Control port has only 4 pins but Data port has 8 pins. Control port pin bit will be 4-lower bit, called C0,C1,C2 and C3. Then interface with EX-01 board, Control port will display only 4 lower bits and some bit will invert. They are C0,C1 and C3 bit.

Sample experiment-3

Status port reading value with INP instruction of Visual BASIC via P-Board

Material :

1. P-Board Parallel port Interface board x 1
2. EX-01 board x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher
4. IDC-10 cable x 1

Procedure

3.1 Connect P-board with PCís parallel port and connect P-board with EX-03 at DIP switch side by IDC-10 cable at DATA BUS connector. See figure-7

3.2 Open Visual BASIC.

3.3 Build form, place Command button and Textbox following figure-8. The **`Text1`** box would be to display switch data reading.

3.4 Add *Inpout32.dll* file into the program for using INP and OUT instruction.

3.5 Write the code below for **`Command1`**

```
        Text1.Text  =  Inp(&H379)
```

3.6 Run the test program. Change the DIP switch value and see the result operation at **`Text1`** box.

After changing the DIP switch value from 3 to 7 position, The read value will change but the changing of 0 to 2 position will not effect because the Status por twill provide only 5-upper bit S3 to S7.

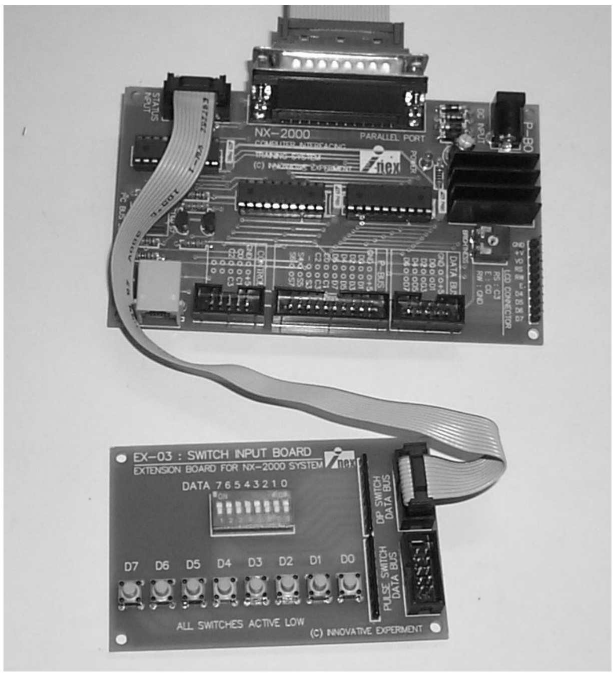


Figure-7 Interface P-Board with EX-03 board

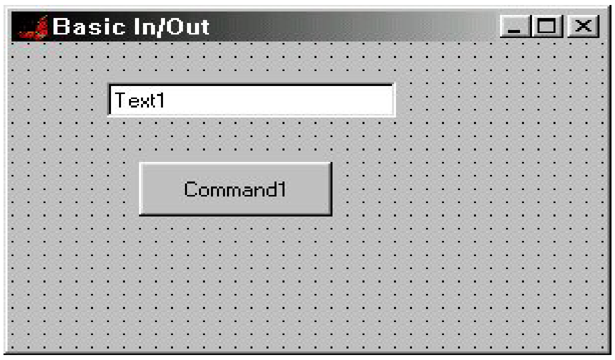


Figure-8 Placing Command button and Textbox

3.7 Reading the value by Text1.Text = Inp(&H379) command, it cause the value difficult to understand. The suitable value format is Hexadecimal. Edit program by adding Hex$ command as :

```
        Text1.Text  =  Hex$(Inp(&H379))
```

3.8 However the reading value will still contain the unused bit which includes S2,S1 and S0 bit. It causes each reading to be incorrect. Try editing the program again. Add Logic instruction to solve this error. The edit program is :

```
        Text1.Text  = Hex$(Inp(&H379)  And  &HF8  )
```

3.9 From editing this program in step 3.8, value in S2, S1 and S0 bit is ì0î. Because AND operation all bit with ì0î value. Then F8 value in binary format is 11111000. Now the value in Text1 box is real switch data.

3.10 However the result from step 3.9 still has some error. S7 bit value must invert. Then add program below to fix this error :

```
        Text1.Text  = Hex$(Inp(&H379)  And  &HF8  Xor  &H80)
```

3.11 In reading value, user must click Command1 button every reading. Add some program below to automatic reading. It use Timer. See figure-9

```
        Private  Sub  Timer1_Timer()
        Text1.Text  = Hex$(Inp(&H379)  And  &HF8  Xor  &H80)
        End  Sub
```

3.12 Change the Interval value of Timer1 in Property to 500, see figure-10.

When program run 0.5 second, it will be jump to Timer1_Timer() subroutine for read switch input. If changing happen, the data in the box will change immediately. To make it faster, change the Interval value of timer to reduce the needed time.

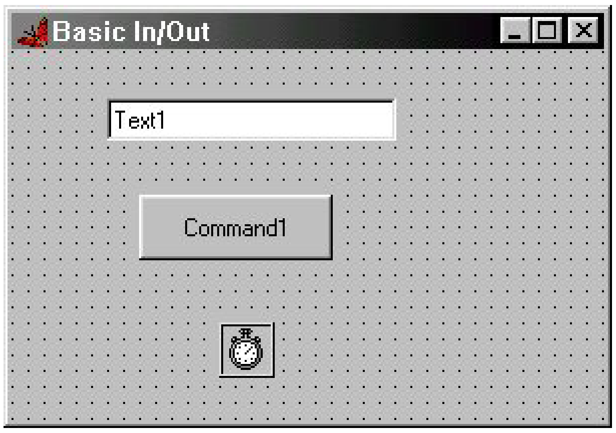
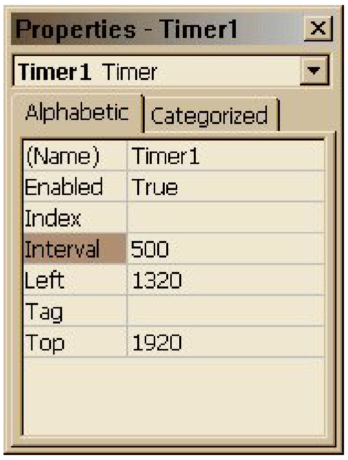


Figure-9 Add timer for continuous operation.



Properties - Timer1

Timer1 Timer

Alphabetic | Categorized

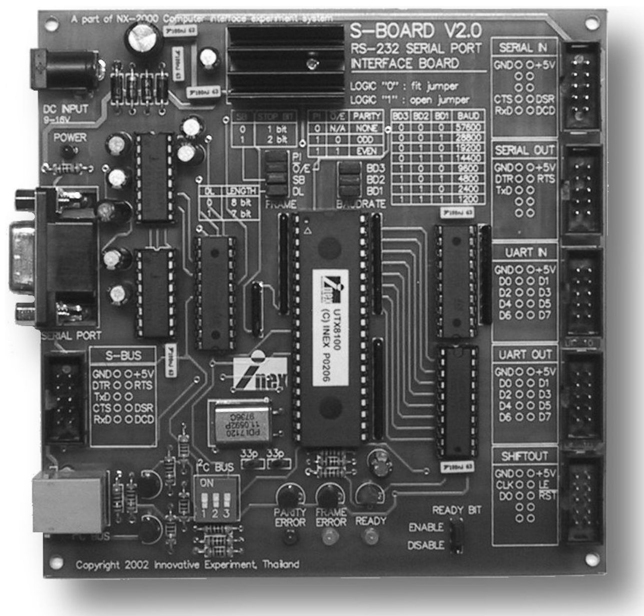
(Name)	Timer1
Enabled	True
Index	
Interval	500
Left	1320
Tag	
Top	1920

Figure-10 Change Interval value in Timer1 to select speed of program response.

S-Board V2.0

Serial port Interface Board

(C) Innovative Experiment Co.,Ltd.



Features :

- ● Interface with RS-232 serial port
- ● On-board RS-232 driver
- ● UART circuit function by custom controller ; UTX8100
- ● Selectable baud rate and data format by jumpers
- ● LED display of status and errors
- ● On-board I^{2}C bus converter circuit
- ● Extension connector for DATA BUS, UART IN, UART OUT, SHIFT OUT and S-BUS For connecting EX-series board

Circuit description

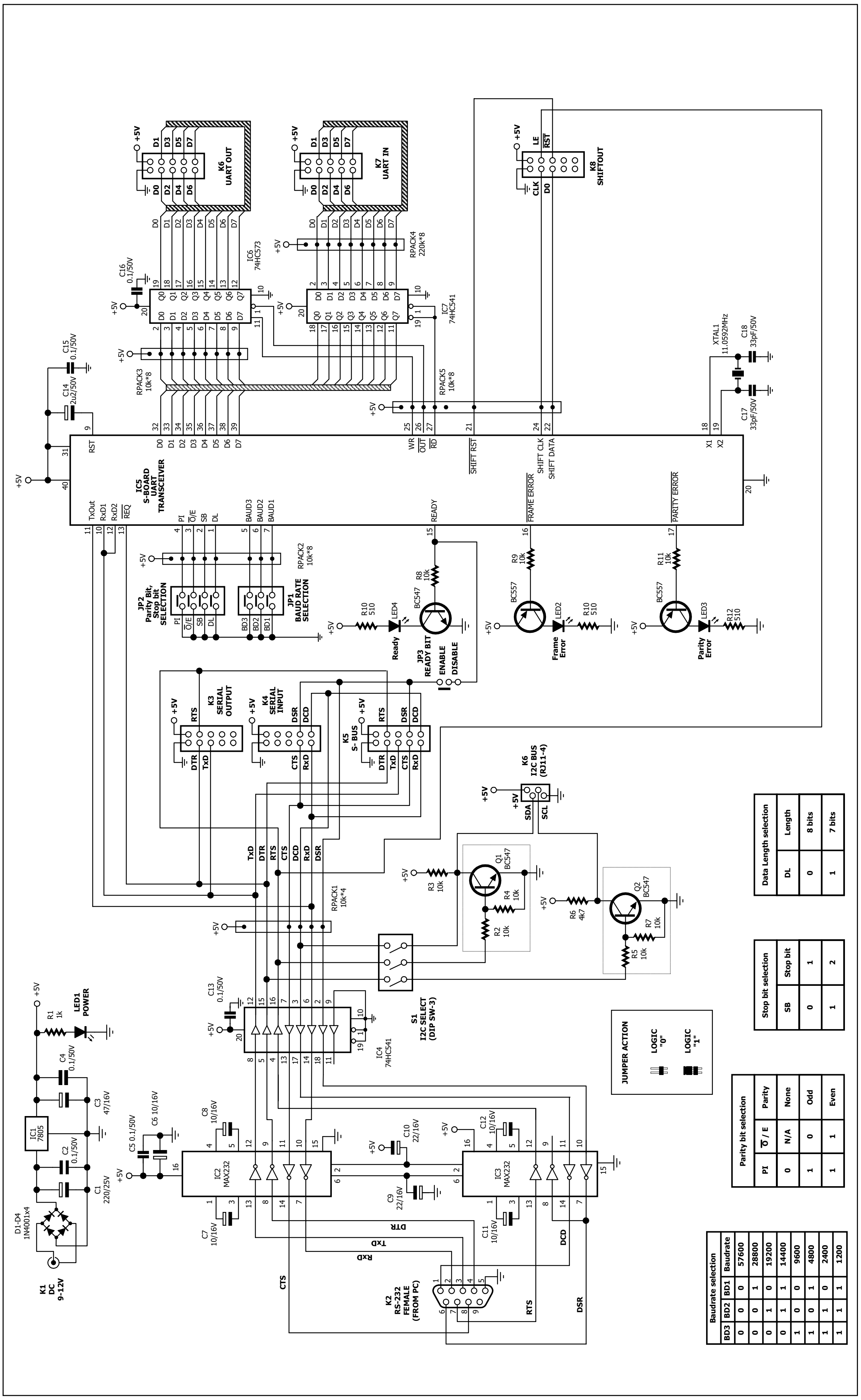
S-Board V2.0 board is computer serial port interface experiment board. The complete shcematic diagram shows in the figure 1. Serial port signal level from computer is following in RS-232 standard. They are ±3V to ±12V. S-board has 2 RS-232 transceiver ICs ; MAX232 for convert the level to TTL level. It helps interfacing with another device more suitable.

S-BUS connector

All serial port signals that via MAX232 ICs are sent to buffer IC ; 74HC541 for current driver and protect the computerís port from error. All buffered serial port signal are connected to S-BUS connector.

SERIAL OUTPUT connector

Output signals of serial port include TxD, DTR and RTS. All 3 signals are connected to SERIAL OUTPUT connetor for driving output device.



Baudrate selection			
BD3	BD2	BD1	Baudrate
0	0	0	57600
0	0	1	28800
0	1	0	19200
0	1	1	14400
1	0	0	9600
1	0	1	4800
1	1	0	2400
1	1	1	1200

Parity bit selection		
PI	O̅ / E	Parity
0	N/A	None
1	0	Odd
1	1	Even

Stop bit selection	
SB	Stop bit
0	1
1	2

Data Length selection	
DL	Length
0	8 bits
1	7 bits

Figure 1 S-Board V2.0 Serial port interfsace board schematic diagram

SERIAL INPUT connector

Output signals of serial port include DCD, CTS, RxD and DSR. All signals are connected to SERIAL INPUT connector for receiving the external digital input signal.

I^{2}C cus jack

S-Board V2.0 board provides another extension bus. It is called I^{2}C bus. The I2C bus signal includes SCL and SDA. But all serial port signal are not bi-directional. Then must use 3 psignal to generate I2C bus signal

For SDA signal use RTS for transmitting and DCD for receiving. For SCL will use DTR signal to drive clock via transistor Q_2

All signal of I2C bus include +5V and Ground are assigned to I2C bus jack in 6P4C modular jack.

UART circuit

For serial communication experiment completely S-Board V2.0 board provides a UART circuit. The heart of this circuit is UTX8100 UART custom controller. Its function similar popular UART chip. UTX8100 can interface in asynchronous serial data communication. The signal pin is TxD and RxD.

TxD signal will send the serial data to UTX8100 via RxD1 and RxD2 pin. UTX8100 will convert to 8-bit parallel data otu to D0 to D7 pin. All output sigal will pass to Latch driver 74HC573 and out to UART OUT connector for interface external output devices.

RxD signal will receive the serial data from TxOut of UTX8100 when trig with logic ë0í from DTR signal at REQ pin of UTX8100. The input data will get from D0 to D7 pin. They are latched from UART IN connector. Data in from UART IN will buffer via 74HC541. UTX8100 will convert 8-bit parallel data input to serial data and send to RxD.

UTX8100 can select baudrate 8 steps as : 1200, 2400, 4800, 9600, 14400, 19200, 28800 and 57600 bit per second by JP1 jumper that connected to BAUD1-BAUD3 pin of UTX8100

Except baudrate setting, UTX8100 can set data bit, parity checkinh and set stop bit by selected jumper. In case detect the error, UTX8100 will inform by error output pin ; *Parity Error (PE) and Frame Error (FE). All error output pin will out logic '0'. If connected the display circuit active low, will see the operation.*

In case no data in transmit register of UTX8100, at Ready pin of UTX8100 will appear logic ë1í signal. If connected the display circuit active high, will see the operation to report UTX8100 has not data in transmit buffer and ready to send.

SHIFTOUT connector

S-Board V2.0 can expand the output port with synchronous data comminucation via SHIFT OUT. It includes SHIFT CLK and SHIFT DATA signal from UTX8100 controller. Both signal are sent to SHIFTOUT connector in name CLK (clock) and D0 (serial data). LE signal or Latch enable receive signal from the serial portís RTS signal. Last, RESET (RST) is SHIFT RST of UTX8100. The suitable expansion board for this connector is EX-09 board. Experimenters can expand output ports 8 to 64 bit from 4 signal and 4 EX-09 boards.

Sample experiment -1
Serial to Parallel data conversion

Material :

1. S-Board V2.0 Serial port Interface board	x 1
2. EX-01 board	x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher	
4. IDC-10 cable	x 1

Procedure

Define serial data protocol between compuerís serial port and UART controller on S-Board V2.0 to 57,600 baud, 8-bit data, no parity and a stop bit.

1.1 Connect IDC-10 cable from UART OUT connector on S-Board to DATA BUS1 connector of EX-01 board.

1.2 Set baudrate jumpers on S-Board to select 57600. By BD1, BD2, BD3 as ì0î

1.3 Set PI, SB jumper as ì0î to select no parity and one stop bit.

1.4 Set DL jumper as ì0î to select 8-bit length.

1.5 Open Visual BASIC. Build form and edit control following the figure P1-1

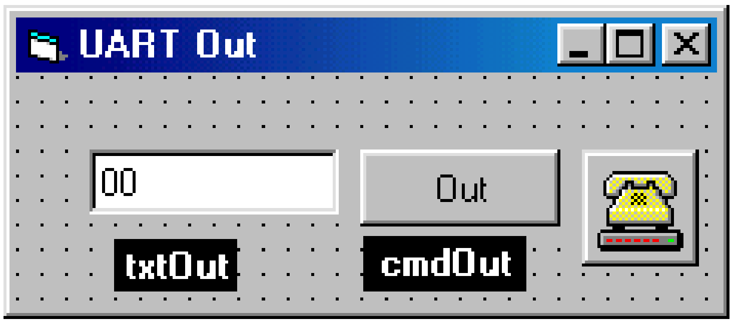


Figure P1 Shows the UART out experiment program. It used to receive data from serial port and out to UART OUT of S-board.

1.6 Write program for **Form_Load** event to define data communication format as :

```
Private Sub Form_Load()
     MSComm1.CommPort = 1
     MSComm1.Settings = "57600,n,8,1"
     MSComm1.PortOpen = True
End Sub
```

1.7 Write program for **cmdOut_Click** event as :

```
Private Sub cmdOut_Click()
     MSComm1.Output = Chr(Val("&H" & txtOut.Text) Mod 256)
End Sub
```

*The **txtOut** data format is hexadecimal but data that sent via **MSComm. Output** must as character. Then convert the data to decimal in range 0 to 255 and use* **`Chr()`** *function to convert to character.*

Sample experiment -2
Parallel to Serial data conversion

Material :

1. S-Board V2.0 Serial port Interface board x 1
2. EX-03 board x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher
4. IDC-10 cable x 1

Procedure

2.1 Connect IDC-10 cable from UARTIN connector on S-Board to DATA BUS of push-button switch on EX-03 board.

2.2 Set baudrate jumpers on S-Board to select 57600. By BD1, BD2, BD3 as ì0î

2.3 Set PI, SB jumper as ì0î to select no parity and one stop bit.

2.4 Set DL jumper as ì0î to select 8-bit length.

2.5 Open Visual BASIC. Build form and edit control following the figure P2

2.6 Write program for Form_Load to define data communication format as :

```
Private Sub Form_Load()
    MSComm1.CommPort = 1
    MSComm1.Settings = "57600,n,8,1"
    MSComm1.PortOpen = True
    MSComm1.DTREnable = False
End Sub
```

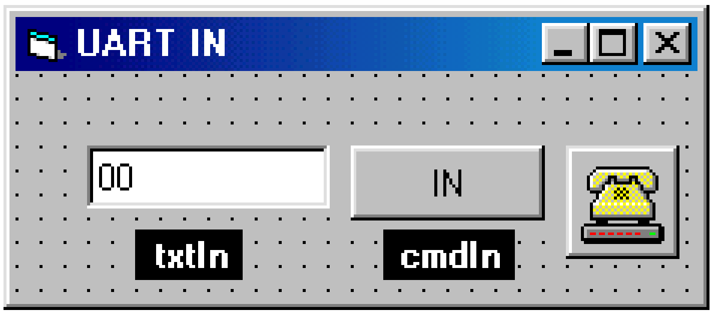


Figure P2 Shows the UART in experiment program. It used to receive data from UART IN of S-board and send with serial data to computer

2.7 Write the delay routine as :

```
Private Sub Delay()
Dim a As Single
    a = Timer + 0.01
    Do While a > Timer
        DoEvents
    Loop
End Sub
```

2.8 Write program for **cmdIn_Click** as :

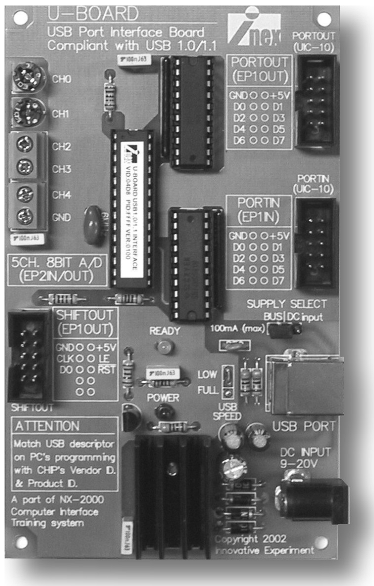
```
Private Sub cmdIn_Click()
Dim tmp As String
    MSComm1.DTREnable = True
    Delay
    MSComm1.DTREnable = False
    Delay
    If MSComm1.InBufferCount > 0 Then
        tmp = MSComm1.Input
        txtIn.Text = Hex(Asc(tmp))
    End If
End Sub
```

Receiving data of UART on S-Board must send signal to REQ pin of UTX8100 before. REQ pin will connect with DTR signal from serial port. Because the modern computers speed are very fast, it cause the UART chip may be cannot detect data in time. The delay routine will be very important. Program will poll for receive data at RxD pin by verify serial port buffer. If data is stored in buffer, program will exit the loop and read buffer data to shows at ***txtIn*** *box. If require to read data automatic, can use* ***Timer*** *control to assist.*

U-Board

USB port Interface Board

(C) Innovative Experiment Co.,Ltd.



Features :

- Interface with USB port
- Support USB V1.0/1.1
- Powered by external or USB port (selection by jumper settings)
- Extension connector for PORTOUT, PORTIN and SHIFT OUT For connecting EX-series board
- 8-Bit A/D circuit 5 channels (2 on-board and 3 for external voltages)

Circuit description

U-Board schematic diagram is shown in the figure 1. The main component is Microchipís USB pre-programmed microcontroller ; PIC16C745. This chip can conntect to USB port directly. In this schematic connect USB port pin series with resistors for current limited protection. Jumper JP1 **POWER SUPPLY** is used to select the power supply for U-Board from USB port or external If not drive the exrernal devide over 100mA, experimenters can use power supply +5V from USB port. If require more current, must use external power supply

I/O port of U-Board prepares 4 ports; **PORTIN** (input port), **PORTOUT** (output port), **SHIFTOUT** (synchronous expansion port) and **A/D input** for ADC module within USB microcontroller. 2 first digitlal port are assigned to Endpoint 1 and A/D is Endpoint 2. Interface with USB microcontroller will do via 74HC573. In sending data to PORTOUT, USB micro will send data on D0 to D7 and Latch signal to 74HC573. The data will appear on PORTOUT connector. In receiving data from PORTIN will appear when USB micro send RD (read) signal to 74HC541. Data from PORTIN connector will appear at D0 to D7 of USB microcontroller

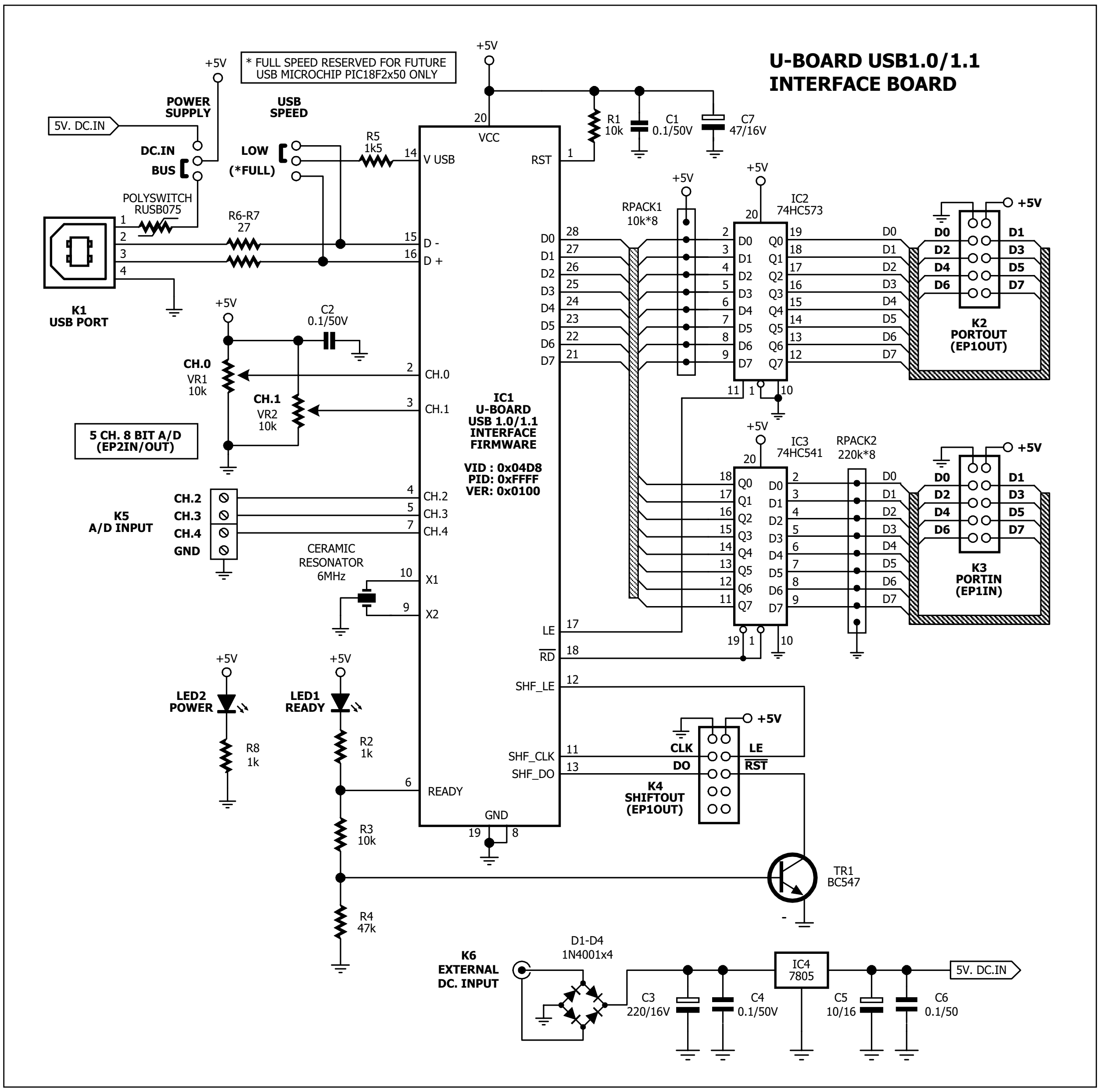


Figure 1 Schematic diagram of U-Board

USB microcontroller has 5-ch. 8-bit analog to digital converter module. In this U-Board connect 2 of variable resistor for on-board variable voltage source and connect 3 terminal blocks for receiving external analog voltage.

U-Board is config to low speed HID class USB device. It cause cannot connect more U-board on the bus.

Preparation

(1) Install U-Board driver

(2) Install HIDcomm control

(3) Write the application program by Visual BASIC or another for interface with U-Board via HIDcomm control.

The important of this experiment is How to Windows know U-Board. The main linker is **U-Board driver and HIDcomm**

The step from here is very important. Experimenters must do becareful.

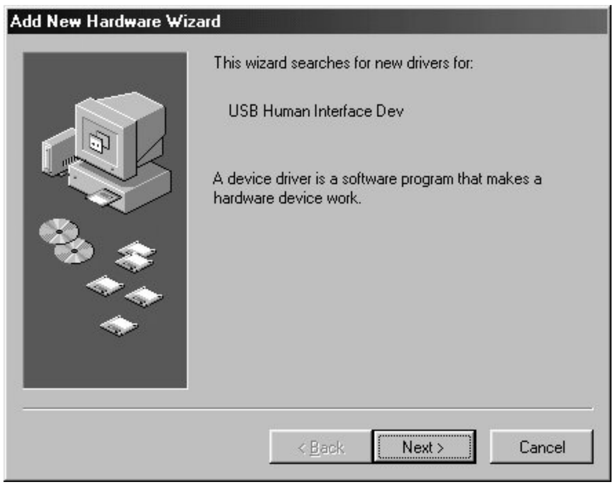
Install U-Board driver

U-Board is config to HID (Human Interface Device) type then can use HID driver within Windows for connection. The install step is :

(1) Select U-boardís power supply to USB port.

(2) Connect USB cable between U-Board and USB port.

(3) Fust time connection, Windows will detect U-board as HID USB device Click Next button

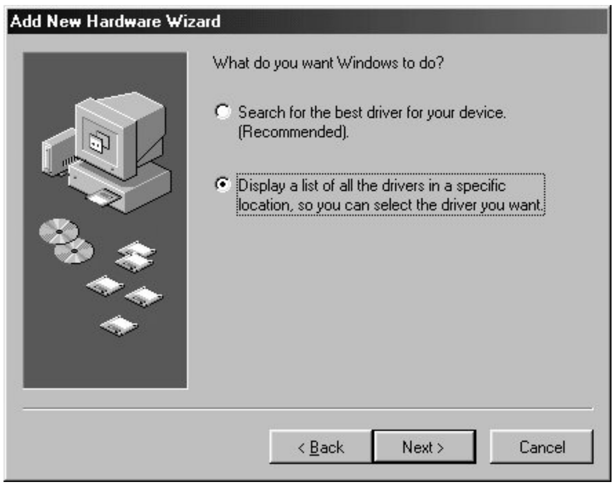

Add New Hardware Wizard

This wizard searches for new drivers for:

USB Human Interface Dev

A device driver is a software program that makes a hardware device work.

(4) Select to Display a list of all the drivers in a specific location.... Click Next button.

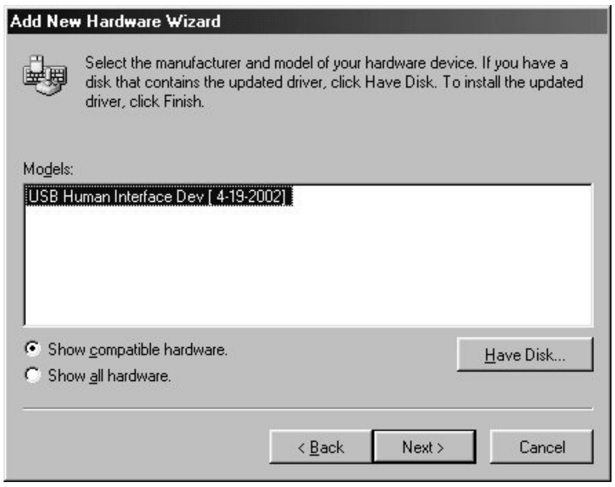

Add New Hardware Wizard

What do you want Windows to do?

( ) Search for the best driver for your device. (Recommended).

(●) Display a list of all the drivers in a specific location, so you can select the driver you want.

(5) Select USB Human Interface Device. Click Next button.


Add New Hardware Wizard

Select the manufacturer and model of your hardware device. If you have a disk that contains the updated driver, click Have Disk. To install the updated driver, click Finish.

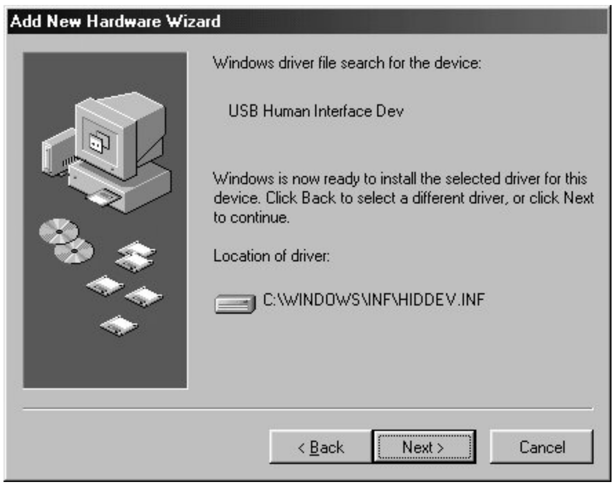
Models:

USB Human Interface Dev [4-19-2002]

(●) Show compatible hardware.

( ) Show all hardware.

(6) The wndow below will appear. Click Next button to start installation.


Add New Hardware Wizard

Windows driver file search for the device:

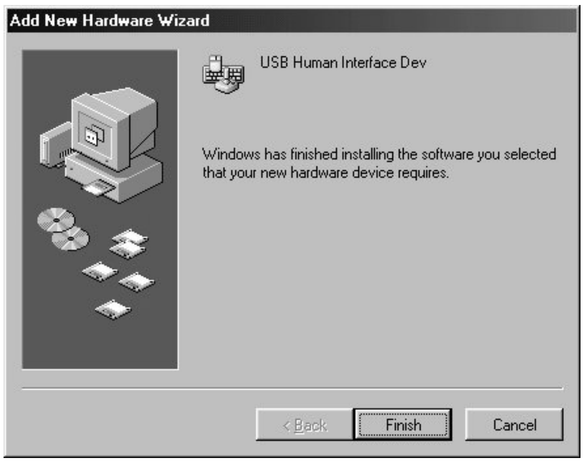
USB Human Interface Dev

Windows is now ready to install the selected driver for this device. Click Back to select a different driver, or click Next to continue.

Location of driver:

C:\WINDOWS\INF\HIDDEV.INF

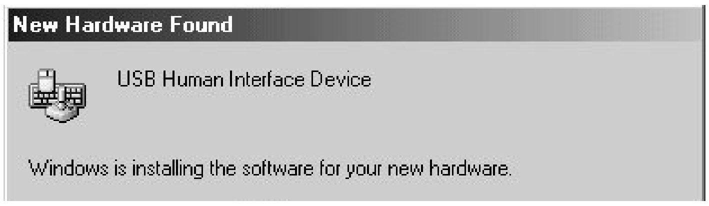
(7) Click Finish button. Installation is complete.


Add New Hardware Wizard

USB Human Interface Dev

Windows has finished installing the software you selected that your new hardware device requires.

For Windows ME : The first time connection of U-Board will see the dialoque box infrom HID USB device detected following the figure below. After that syytem will install driver automaticcally and not appear the dialog box from step (1)


New Hardware Found

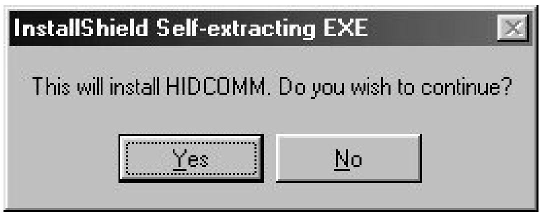
USB Human Interface Device

Windows is installing the software for your new hardware.

After driver installation complete, next step is making the application software ti interface U-Board. In U-Board programming, suggess to use API function. The API function is internal function of HID. Normally API programming will be very complex and require more programming skill. Microchip help developers by make the custom control to support.This control is called **HIDComm.** U-Boardís experimental program use this control in same.

HIDComm installation

(1) Run Microchip HIDComm ActiveX setupex.exe file for installation. This file will be contained in INEXís Computer interface CD-ROM. The installation window will appear.


InstallShield Self-extracting EXE

This will install HIDCOMM. Do you wish to continue?

(2) Accept all installation instruction untill finish.

Sample experiment-1
Interface U-Board with USB by HIDComm

Material :

1. U-Board USB port Interface board with USB cable x 1

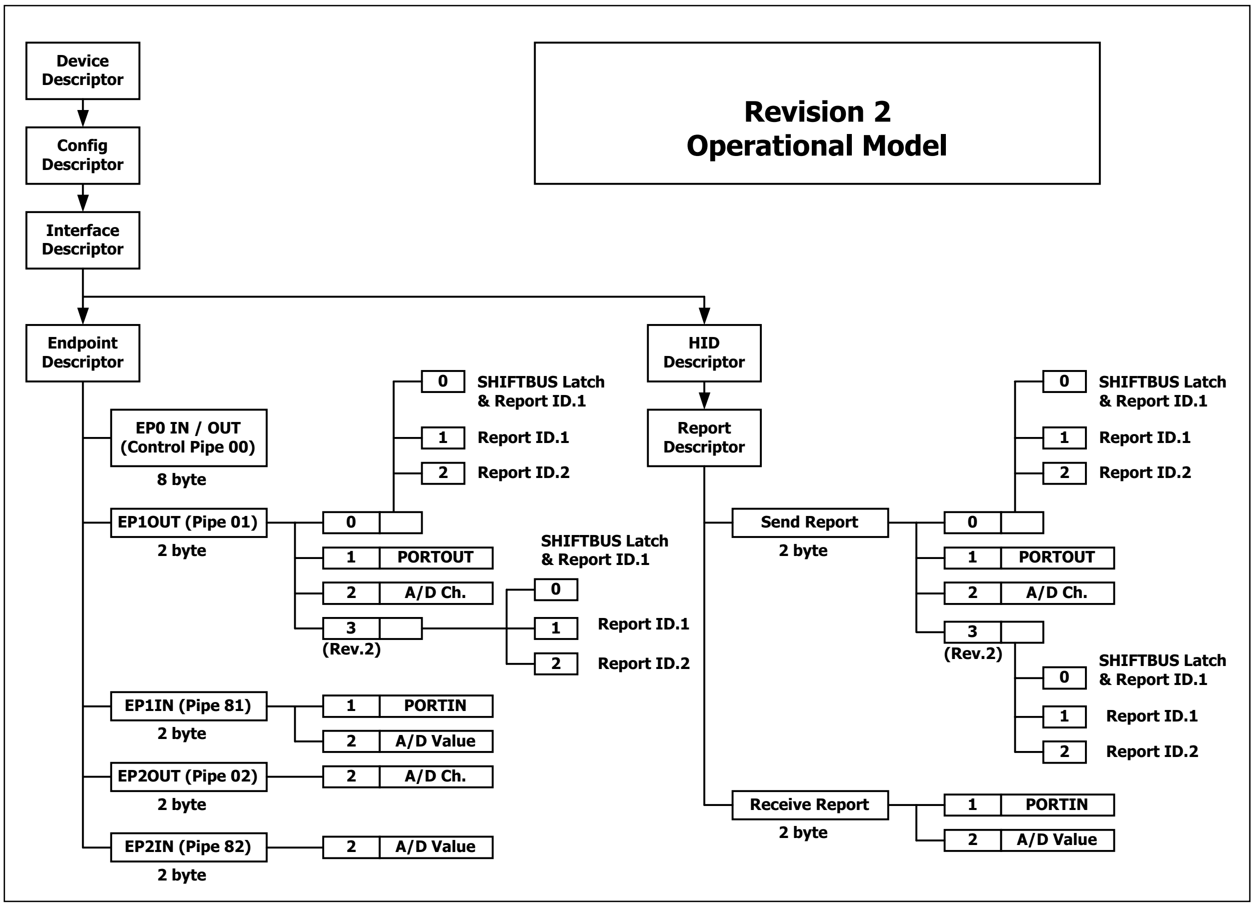
2. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

Concept :

U-board data transfer protocol

The figure P1-1 shows the software operation model of U-Board. U-Board is designed as HID class USB device. The Interface descriptor will use HID desicriptor to work with Endpoint descriptor. The operation format as :

● Transfer data 2 byte/packet. First byte is Report ID. Second byte is data.



The figure P1-1 shows the software operation model of U-Board

● After receive data from computer, U-Board will collect Report ID of receiving data for prepare to send same Report ID back to computer when request.

● Cannot select Report ID in transmittion data to computer. This is limitation of USB1.0

● Transfering data with any port are defined by Report ID as :

Report ID 0 Define the Report ID of transmittion to computer. There is 3 values as :

"0" Set Report ID of transmit data as 1 and latch data to SHIFT OUT port on U-Board

ì 1 î Set Report ID of transmit data as 1

"2" Set Report ID of transmit data as 2

Report ID 1 Receive data from computer and pass to PortOut and read Port In data for sending back to computer.

Report ID 2 Receive data for select A/D input and send the conversion result.

● After receive data 1 packet, must delay at least 71 millisecond for U-Board initial operation before send data to computer.

Using HIDComm control with U-Board

U-Board connection

Must connect HIDComm with U-Board before transfer data. Experimenters can use **Browse** function to select the USB device. The sampl;e code shows below :

```
Private Sub cmdSelectDevice_Click()
    HIDComm.Browse
    HIDComm.Connect
End Sub
```

Another method, define the properties of HIDComm in the program to match with USB device in the program. Connect properties has 6 items as :

```
HIDComm.MatchManufacturer = "Innovative Experiment"
HIDComm.MatchProduct = "U-Board USB1.0/1.1 Interface"
HIDComm.MatchSerial = 0
HIDComm.MatchPID = 4095
HIDComm.MatchVID = 1240
HIDComm.MatchVersion = 256
HIDComm.Connect
```

HIDComm event

All HIDComm events can indicate the interface status. It devided 2 groups as

1. **Connection status**

   ConnectFailure

   ConnectSuccess

   Disconnected

2. **Transfer status**

   ReadFailure

   ReadSuccess

   WriteFailure

   WriteSuccess

Procedure :

1. Connect U-Board with USB port

2. Place some control following the figure P1-2

3. Write the code for any event as :

```
Private Sub cmdConnect_Click()
     HIDComm.MatchManufacturer = "Innovative Experiment"
     HIDComm.MatchProduct = "U-Board USB1.0/1.1 Interface"
     HIDComm.MatchSerial = 0
     HIDComm.MatchPID = 4095
     HIDComm.MatchVID = 1240
     HIDComm.MatchVersion = 256
     HIDComm.Connect
End Sub

Private Sub Form_Load()
     HIDComm.Browse
     HIDComm.Connect
End Sub
```

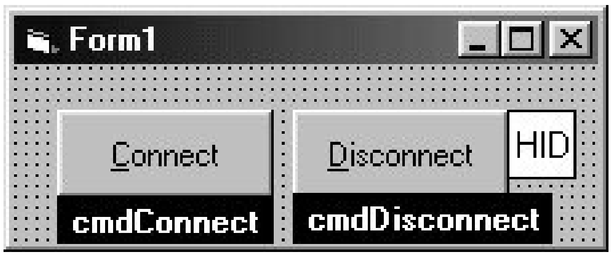
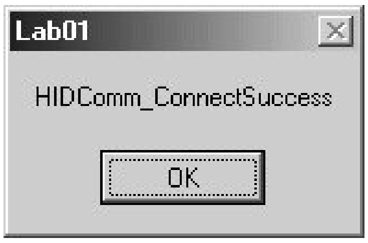


Figure P1-2 The U-Board with HIDComm connection program's window

```
Private Sub cmdDisconnect_Click()
    HIDComm.Disconnect
End Sub

Private Sub HIDComm_ConnectFailure(ByVal Status As Long)
    MsgBox "HIDComm_ConnectFailure"
End Sub

Private Sub HIDComm_ConnectSuccess(ByVal Status As Long)
    MsgBox "HIDComm_ConnectSuccess"
End Sub

Private Sub HIDComm_Disconnected(ByVal Status As Long)
    MsgBox "HIDComm_Disconnected"
End Sub
```

In running this probram, program window (figure P1-2) will appear. Click Connect button, The dialog box (figure P1-3) of connection between U-Board and HICComm will appear.

Full sourcecode can see in LAB01.VBP file in USB_port/Lab/Lab01 folder within INEXís Computer Interface CD-ROM.


Lab01

HIDComm_ConnectSuccess

Figure P1-3 The U-Board with HIDComm connection status dialog box

Sample experiment - 2
PORTOUT experiment

Material :

1. U-Board USB port Interface board with USB cable   x 1
2. EX-01 board   x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher
4. IDC-10 cable   x 1

Concept :

PORTOUT of U-Board use ReportID 1 for interfacing. This component can send data each 2 bytes. The first byte is Report ID that HIDComm control willl transmit aitomatically. The second byte is data that outs to PORTOUT connector.

Procedure :

2.1 Connect EX-01 with PORTOUT connector of U-Board.

2.2 From program in experiment -1, add TextBox and CommandButtonfor transmitting data to U-Board following the figure P2-1

2.3 Write program for **cmdWrite_Click** event as :

```
Private Sub cmdWrite_Click()
Dim Buffer(0) As Byte
    Buffer(0) = CByte("&H" & txtOut.Text)
    HIDComm.ReportID = 1                  ' Send Report ID
    HIDComm.WriteTo Buffer, 1             ' Transmit data to U-Board
End Sub
```

Sending data through HIDComm control use **`WriteTo`** *command. Parameter of this command must as Array variable and following data byte. From the experiment code, data sending is width 1 byte thus declare one Array variable start at 0. After that get data from* ***txtOut*** *to convert to numerical and store at Array variable. Define Report ID value and send data to output.*

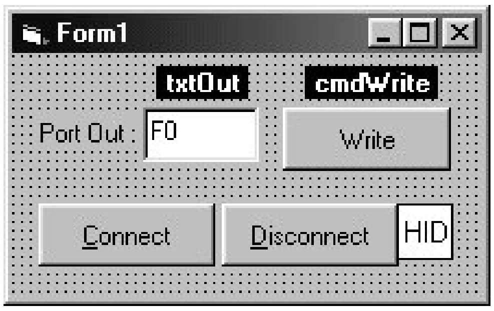


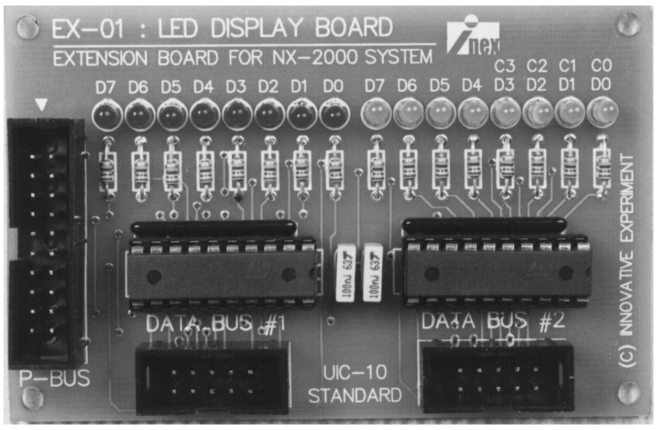
Figure P2-1 The PORTOUT program window

Experiment result : *When program is running, initialize interface between U-Board with HIDComm happen before. Experimenters can observe from the interface complete dialog box that appear. Experimenters must click to accept before. After that the experimental program window will appear. Put data into PortOut box and click Write button. That data will send to EX-01 board via PORTOUT connector. LED at EX-01 will show value same at Port Out box in the pogram window.*

Full sourcecode can see in LAB02.VBP file in USB_port/Lab/Lab02 folder within INEXís Computer Interface CD-ROM.

EX-01
16-ch. LED display board

(C) Innovative Experiment Co.,Ltd.


EX-01 : LED DISPLAY BOARD
EXTENSION BOARD FOR NX-2000 SYSTEM
D7 D6 D5 D4 D3 D2 D1 D0
C3 C2 C1 C0
P-BUS
DATA BUS #1
DATA BUS #2
UIC-10 STANDARD
(C) INNOVATIVE EXPERIMENT

Features :

- Two 8-bit LED monitor
- On-board buffer and driver circuit
- Available connector for P-board, S-Board and U-Board

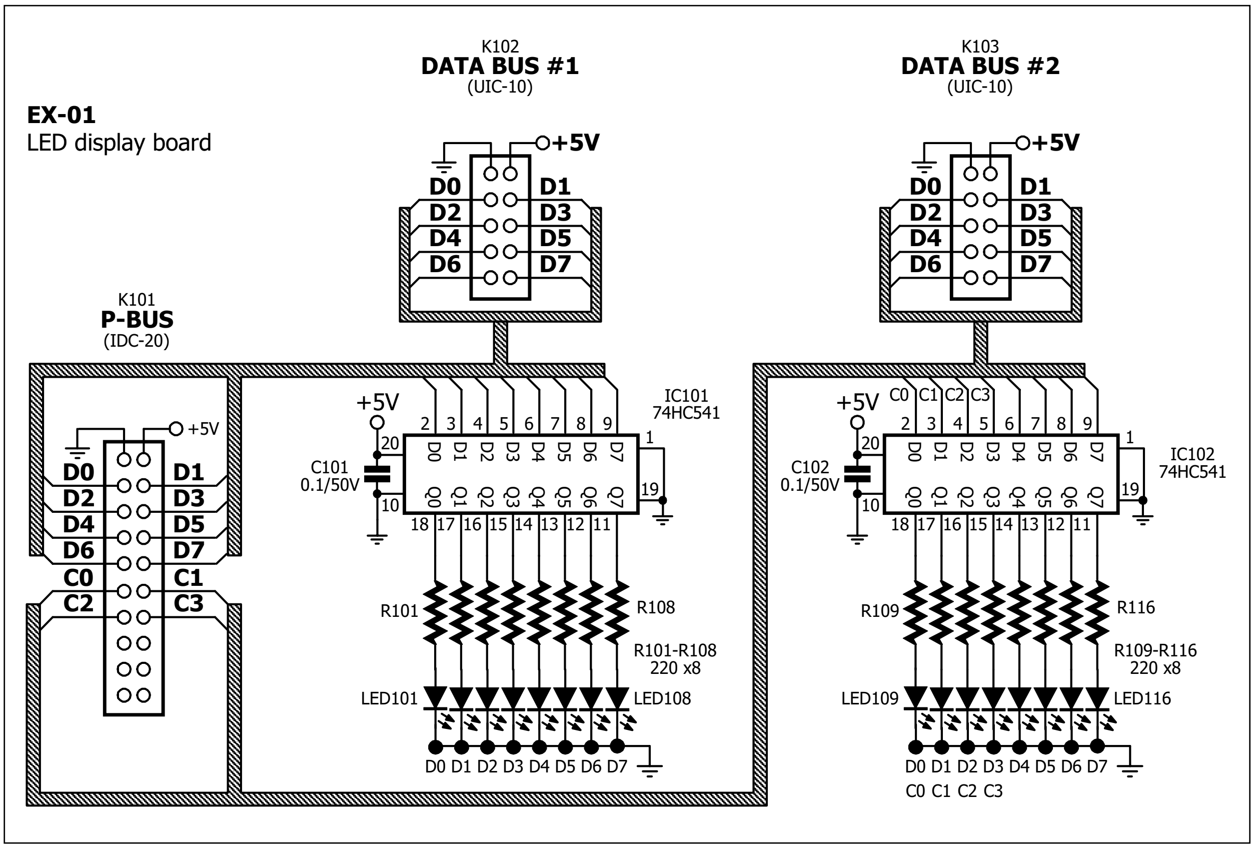


Figure-1 EX-01 schematic

Sample experiment-1

Data port sending value with OUT command of Visual BASIC via P-Board

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-01 board	x 1

3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

4. IDC-10 cable	x 1

Procedure

1. Connect P-board with PCís parallel port and connect P-board with EX-01 by IDC-10 cable at DATA BUS connector. See figure-2

2. Open Visual BASIC.

3. Enter **Project** menu. Select **Add File** to add *INPOUT32.BAS* file (contain in bundle CD-ROM of P-board) into **Project**. At **Project window** will appear *INPOUT32.BAS* file.

4. Place 2 **Command Buttons** into Form1 following figure 3.

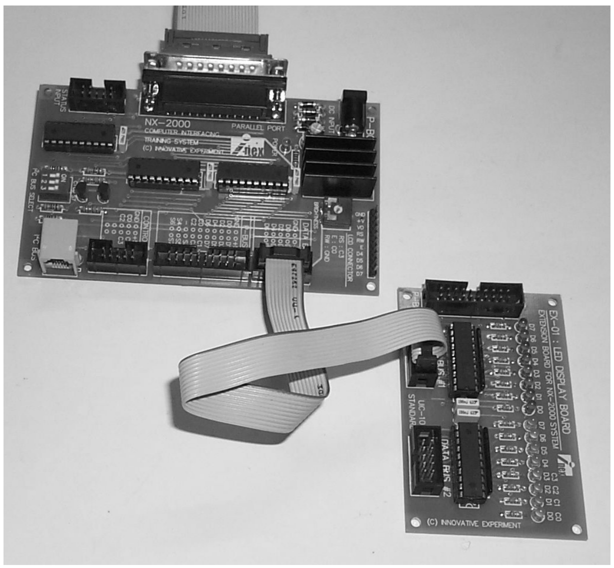


Figure-2 P-board interface with EX-01 board.

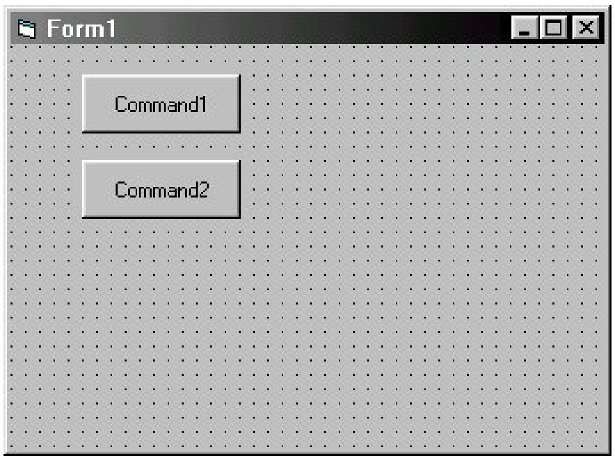


Figure-3 Placing Command Button in Visual BASIC

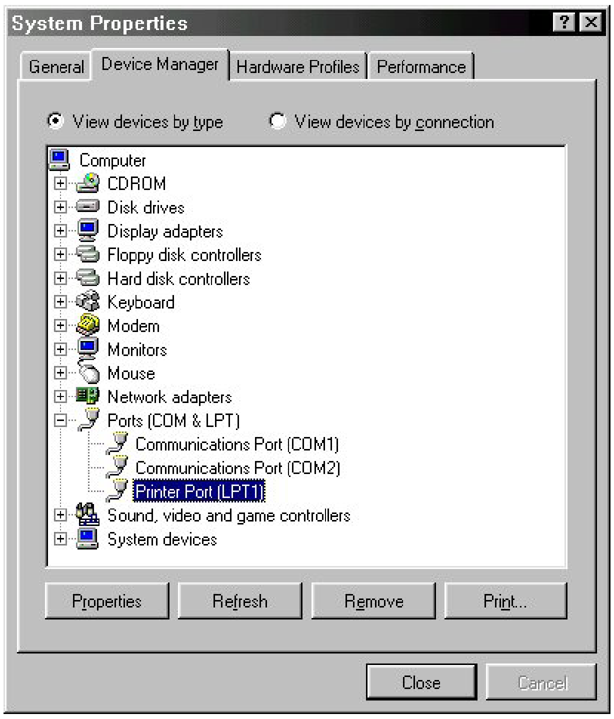
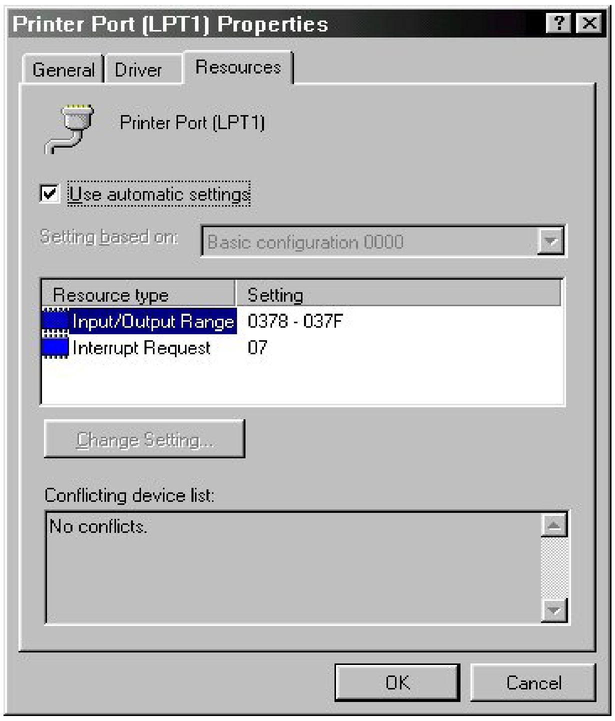


Figure-4 Device Manager window for checking the hardware



Use automatic settings: ✓

Setting based on: Basic configuration 0000

Resource type	Setting
Input/Output Range	0378 - 037F
Interrupt Request	07

Conflicting device list:

No conflicts.

Figure-5 Resource window of Parallel port for report the parallel port

5. Double click at **Command1** to enter **View Code menu**. Write the code below for **Command1_Click** event

```
Private Sub Command1_Click()
Out &H378, &HFF
End Sub
```

6. Double click at **Command2** and write the sourcecode for **Command2_Click** event below :

```
Private Sub Command2_Click()
Out &H378, 0
End Sub
```

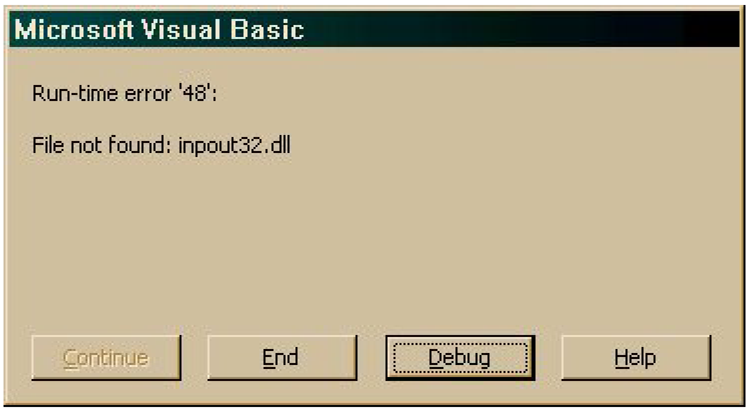
7. Apply voltage for P-Board

8. Run the program. Click**Command1** button. Observe the result at LED on EX-01 board.

8-LEDs on EX-01 board turn on.

9. Click **Command2** button. Observe the result at LED on EX-01 board.

8-LEDs on EX-01 board turn off


Microsoft Visual Basic

Run-time error '48':

File not found: inpout32.dll

Figure-6 Warning message of Inpout32.dll file not founding.

10. If program done not correct, check the parallel port address normally is 378H. Click My Computer and right click to select Property and select to Device Manager. The window in figure-4 will appear. Select Printer Port and click Property button and select to Resource tab. The window in figure-5 will appear. In this window will show the parallel port address the using. First number is DATA port address (0378). If user use different must change.

11. If the warning dialogue box in figure-6 is appeared, it means user do not copy Inpout32.dll fiile into SYSTEM folder of Windows.

12. If all OK, change the value to send to EX-01 such as `OUT &H378,&H55.`

The LED will on and off swap. *Because 55H value is 01010101 in binary. "1" means LED on and "0" means LED off.*

Sample experiment-2

Control port sending value with OUT command of Visual BASIC via P-Board

Material :

1. P-Board Parallel port Interface board x 1
2. EX-01 board x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher
4. IDC-10 cable x 1

Procedure

1. Connect P-board with PCís parallel port and connect P-board with EX-01 by IDC-10 cable at DATA BUS connector. See figure-2

2. Use can use program from experment-1. Edit Command1_Click to

```
Private Sub Command1_Click()
Out &H37A, &HF4
End Sub
```

3. Edit Command2_Click to :

```
Private Sub Command1_Click()
Out &H37A, &HB
End Sub
```

4. Run the program. Try to click Command1 and Command2 button. Observe the operation.

<u>Command2 button is pressed</u>

All LED on EX-01 board turn-off. Because C0, C1 and C3 is invert bit.

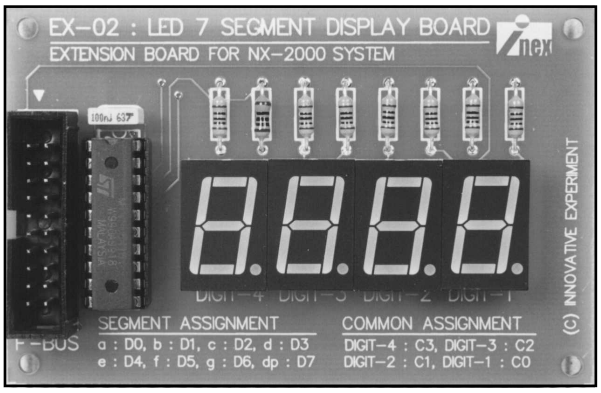
<u>Command1 button is pressed.</u>

LED on EX-01 board will turn-on only 4 positions. Becaise Control port of parallel port has only 4-bits.

Sending data of Control port is similar Data port. The different is Control port address next Data port 2 address. If Data port address is 378H, Control port address will be 37AH. Another different is number of pin. Control port has only 4 pins but Data port has 8 pins. Control port pin bit will be 4-lower bit, called C0,C1,C2 and C3. Then interface with EX-01 board, Control port will display only 4 lower bits and some bit will invert. They are C0,C1 and C3 bit.

EX-02
LED 7 segment display board

(C) Innovative Experiment Co.,Ltd.


EX-02 : LED 7 SEGMENT DISPLAY BOARD
EXTENSION BOARD FOR NX-2000 SYSTEM
inex
DIGIT-4 DIGIT-3 DIGIT-2 DIGIT-1
(C) INNOVATIVE EXPERIMENT
P-BUS
SEGMENT ASSIGNMENT
a : D0, b : D1, c : D2, d : D3
e : D4, f : D5, g : D6, dp : D7
COMMON ASSIGNMENT
DIGIT-4 : C3, DIGIT-3 : C2
DIGIT-2 : C1, DIGIT-1 : C0

Features :

- 4-Digit LED 7 segment display
- On-board buffer and driver circuit
- Active logic ìHIGHî for segment and logic ìLOWî for common control
- Connect with P-Board via P-BUS connector

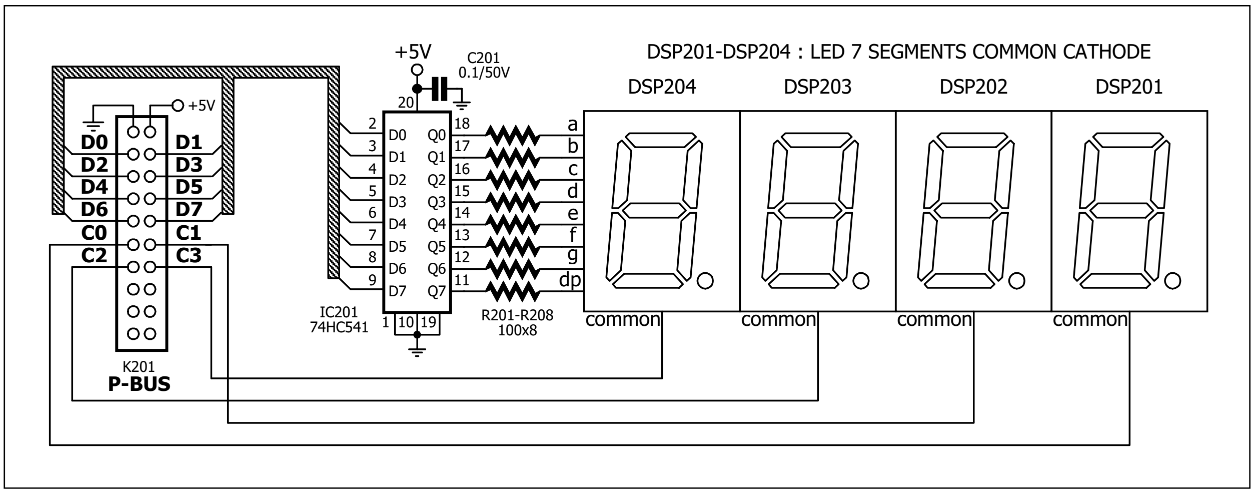


Figure-1 EX-02 schematic

Circuit description

Schematic diagram of EX-02 board show in figure 1. 4 digits of LED 7-segments will connect in multiplex. All same segments will connect together and connect to buffer ICs, 74HC541. Input of 74HC541 are connected to DATA BUS connector that interface with P-Board. The reason is all data pins of parallel port cannot drive LED directly.

LED 7-segments display is common cathode type. It need logic ë1í to drive each segment pin via limited-current resistor. Each common pin of LED are connected to the parallel portís control pin, C0 to C3. User can control LED each digit via this control line.

Because some bit of the parallel portís control pin need invert logic. Itís include C0, C1 and C3. Normally control pin need logic ì0î but this case is different. C0, C1 and C3 need logic ì1î.

Sample experiment-1

Drive LED display single digit

Material :

1. P-Board Parallel port Interface board x 1
2. EX-02 board x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher
4. IDC-20 cable x 1

Procedure

1.1 Make the dispaly data table of LED 7 segments with arrey by wrote the VB code below :

```
Dim Number(0 To 9) As Integer
Private Sub Form_Load()
Number(0) = &H3F
Number(1) = &H6
Number(2) = &H5B
Number(3) = &H4F
Number(4) = &H66
Number(5) = &H6D
Number(6) = &H7D
Number(7) = &H7
Number(8) = &H7F
Number(9) = &H6F
End Sub
```

*From the code, must reserve the memory space with '**Number**' variable. It is decleaed by DIM command at head of the code.*

1.2 Target is after click at Command1 button, LED at right digit will show numerical value and increase value every clicking. The sample code can show below :

```
Dim Index As Integer
Private Sub Command1_Click()
If Index < 10 Then
Out &H378, Number(Index)
Index = Index + 1
Else
Index = 0
End If
Out &H37A, &H5
End Sub
```

*This code declare '**Index**' variable to access the data table, The **Index** value is in range 0 to 9. After that use* `Out` *command to send the data to LED's segment and send logic "0" to common pin of displayed digit.*

However in LED configuration at right digit is connected to C0 pin. This pin will invert logic then send logic “1” to this pin instead. The common control data is 5 in decimal or 0101 in binary to because must turn-off another digit.

1.3 Connect P-Board with EX-02 board by IDC-20 cable via P-BUS connector.

1.4 Apply the supply voltage to both board. Ru the program from step 1.2. Observe the operation at EX-02 board.

1.5 Edit the experiment code to change the display digit by send the value as :

2nd digit (count from the right) display data is &H6

3rd digit display data is &H0

4th digit (The left digit) display data is &HC

Turn-off all digit data is &H4

Turn-on all digit data is &HB

About sample experiment program for Parallel port can see in Lab8.VBP in Parallel_port/Lab/Lab08 folder. All files are contain in INEXís Computer Interface CD-ROM.

Sample experiment-2

Drive LED display in multiplexed mode

Material :

1. P-Board Parallel port Interface board x 1
2. EX-02 board x 1
3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher
4. IDC-20 cable x 1

Procedure

2.1 Write the experiment code below :

```
Private Sub Command3_Click()
Exits = False
Do
Out &H378, Number(1)
Out &H37A, &H5
Call delay
Out &H378, Number(2)
Out &H37A, &H6
Call delay
Out &H378, Number(3)
Out &H37A, &H0
Call delay
Out &H378, Number(4)
Out &H37A, &HC
Call delay
Out &H37A, &H4
Loop Until Exits = True
End Sub
```

This code will diaplay number 1,2,3 and 4 in each digit from the left. After send data to display in each digit must delay to latch the display before send the data to next digit. Human eye cannot see the changing in each digit. Then user will see all LED can work in same time.

For delay routine, user xan adjust the dealy time from scroll bar. Because computer speed in each will be not equal. The sample VB code of delay routine can see below :

```
 Sub delay()
 For i = 1 To HScroll1.Value
 DoEvents
 Next i
End Sub
```

The delay time will depend on scroll bar value. If can select the suitable value (TARGET VALUE), user can define by edit the code at

```
        For i = 1 To Hscroll1.Value
```

to

```
        For  I = 1 To  "TARGET VALUE"
```

After the program run complete in one cycle, loop program will scan the LED again until Exits varaible's value is True. It will be true by pressing Exit button to exit the program.

2.2 Connect P-Board with EX-02 board by IDC-20 cable via P-BUS connector.

2.3 Apply the supply voltage to both board. Run the program from step 1.2. Observe the operation at EX-02 board.

2.4 Try to edit the delay routine to find the best displaying as good as possible.

Computer clock via Parallel port

Procedure

This experiment will be demonstration about digital clock. Use EX-02 board as display board. The timebase will get time value from computerís system clock.

2.5 Make textbox **Text1** for display time value. Use **timer2** for convert time value to LED data display. Write the experiment code below :

```
Private Sub Timer2_Timer()
Text1.Text = Format(Now, "hh:mm:ss")
Index1 = Asc(Right(Text1.Text, 1)) - &H30
Index2 = Asc(Mid(Text1.Text, 7, 1)) - &H30
Index3 = Asc(Mid(Text1.Text, 5, 1)) - &H30
Index4 = Asc(Mid(Text1.Text, 4, 1)) - &H30
End Sub
```

The first line is defining the current time value to **Text1**. *The format is Hour, Nimute and Second.*

Second line is getting only the right letter of **Text1**. *It is first digit of Second value. Get this value by Right command. After that convert this data to ASCII (&H30-&H39 means 0 to 9) For correct number value must subtract the ASCII value with &H30.* ***Index1*** *variable will be 0 to 9 number. User will get this data to open the data table to send to LED 7-segment.*

In 3rd to 5th line of code are similar operation. Each program line operation will be different about data position. It will get the tens of Second, Minute and Hour data to store in ***Index1*** *to* ***Index4*** *variable.*

Displaying method is similar the experiment-1 but not define the arrey address directly, user can use ***Index1*** *to* ***Index4*** *value from calculation to access arrey instead. Re-write the code as :*

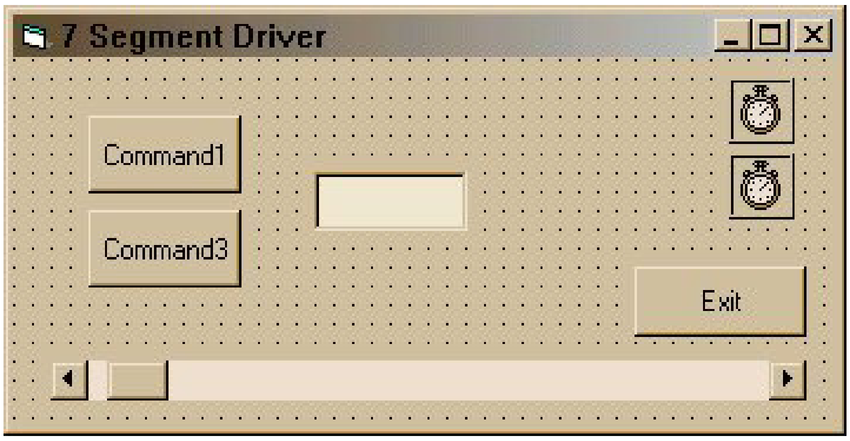


Figure P2-1 Computer clock program's screen

```
Private Sub Command3_Click()
Exits = False
Do
Out &H378, Number(Index1)
Out &H37A, &H5
Call delay
Out &H378, Number(Index2)
Out &H37A, &H6
Call delay
Out &H378, Number(Index3)
Out &H37A, &H0
Call delay
Out &H378, Number(Index4)
Out &H37A, &HC
Call delay
Out &H37A, &H4
Loop Until Exits = True
End Sub
```

The changing of this upper code is putting the variable value into ***Number*** *arrey. In this sample code is get value of Hour, Minute and Second to display on LED. Experimenters can modify program to display only Hour and Minute by editing the code of* **Timer2**. *Interval value of* **Timer2** *is 100 to 1000 millisecond.*

The program window is shown in the figure P2-1 and example program can see in the EX-09.VBP file (See detail in P-board CD-ROM)

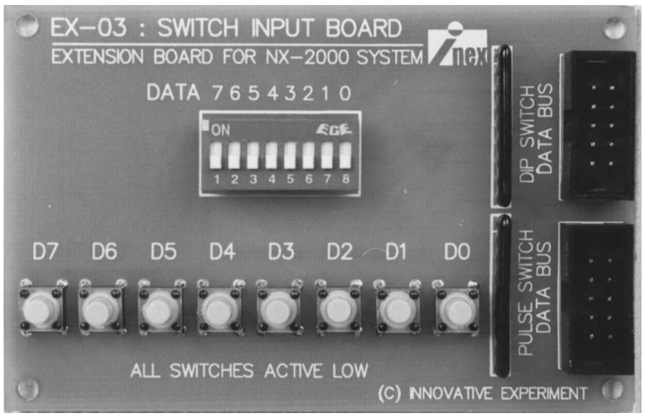
3.2 Connect P-Board with EX-02 board by IDC-20 cable via P-BUS connector.

3.3 Apply the supply voltage to both board. Run the program from step 3.1 Observe the operation at EX-02 board.

About sample experiment program for Parallel port can see in Lab9.VBP in Parallel_port/Lab/Lab09 folder. All files are contain in INEXís Computer Interface CD-ROM.

EX-03
16-ch. Switch input board

(C) Innovative Experiment Co.,Ltd.



Features :

- 8-Ch. DIP switch with pull-up
- 8-Ch. push-button switch with pull-up
- Available connector **ìDATA BUSî**

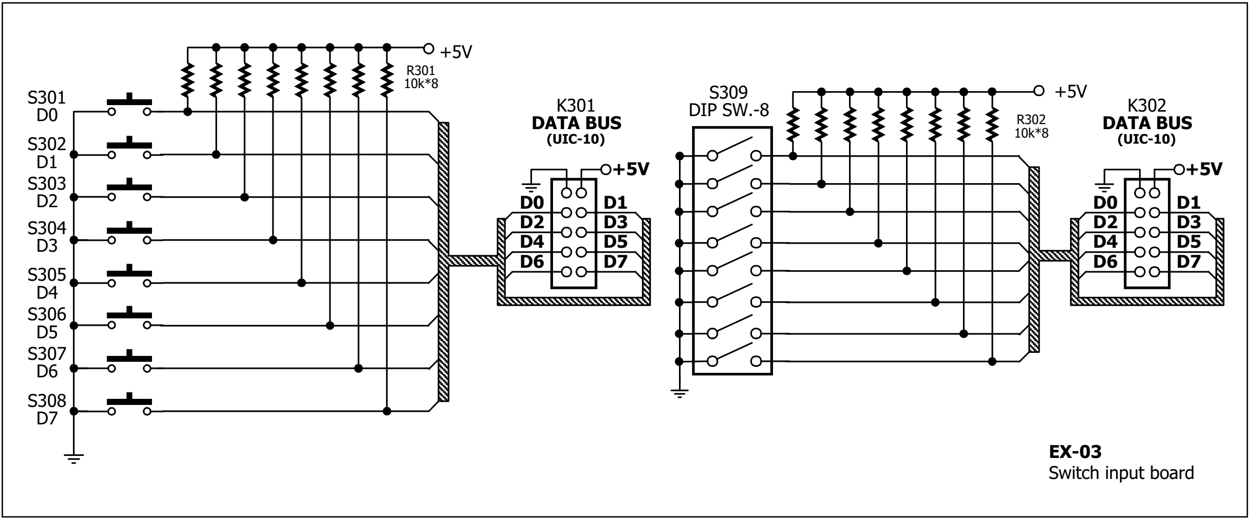
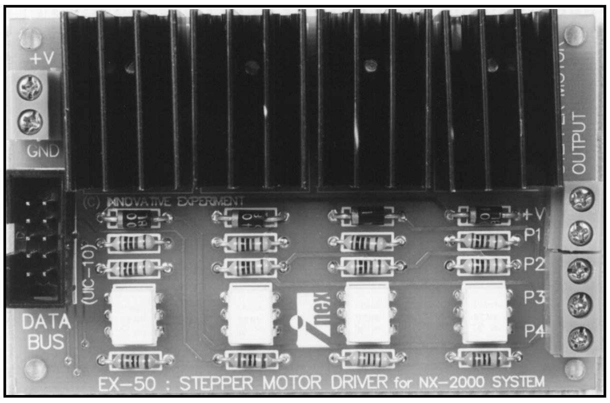


Figure-1 Schematic of EX-03 Switch input board

EX-05
Stepper motor driver board

#8002005 (C) Innovative Experiment Co.,Ltd.


+V
GND
DATA BUS
OUTPUT
P1
P2
P3
P4
(C) INNOVATIVE EXPERIMENT
EX-50 : STEPPER MOTOR DRIVER for NX-2000 SYSTEM

Features :

● Power uni-polar stepper motor 5-24V 1.5A (max)

● Require external power supply

● Ground isolated by opto-coupler

● Connect to P-Board & S-Board via ìDATA BUSî connector

Circuit description

Schematic diagram of EX-05 board show in figure 1. Input signal from DATA BUS will drive infrared LED within opto-coupler IC501-IC504. Output of IC501-IC504 connected with transistor Q501-Q504 BD139 to drive motor. Opto-coupler can active after receive logic ì1î and drive transistor. The coil end will be connect to ground and that phase can active. Stepper motor will spin a step. If all input received the continuous signal, driver circuit will be active and drive motor to spin another step continuous. Experimenters can define the 4-bit input data to control a uni-polar stepper motor. The driving method has 3 method : Full step, Half step and Micro step.

The supply voltage of EX-05 board must separated from the supply source of P-Board . The level can change following motor rating. EX-05 board can use with the supply voltage exceed +30V 1A current.

A parallel port can drive stepper motor 2 channels in same time. Because a motor need 4-line signal. Parallel port has 8-line data. Experimenters can divide to drive 2 motors. However experimenters can use CONTROL pin of parallel port to drive stepper motor. In this case experimenters can drive stepper motor 3-channels

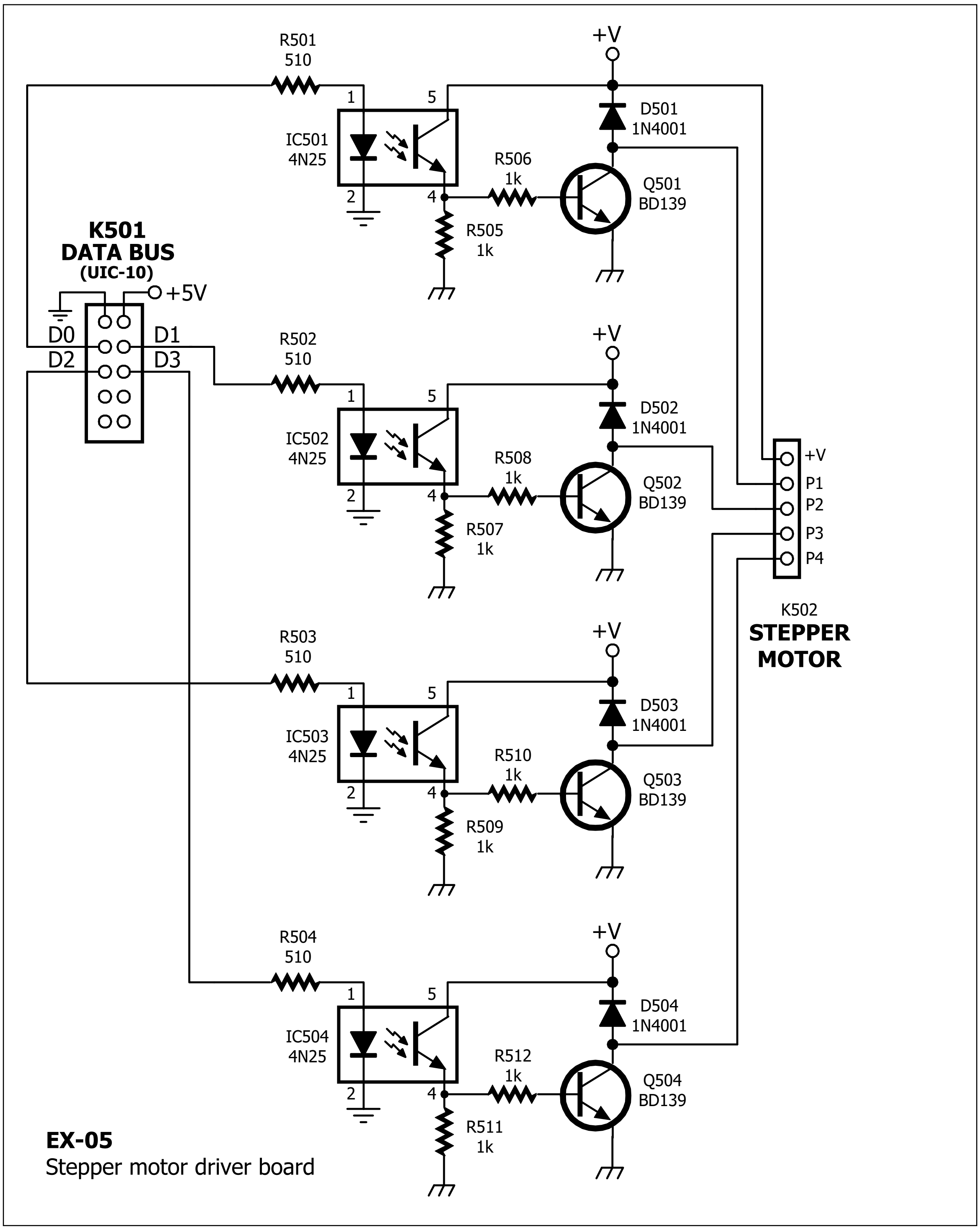


Figure-1 EX-05 Stepper motor driver board schematic

Sample experiment-1

1-Phase Stepper motor driving

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-05 board	x 1

3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

4. +12V 2A external DC power supply fo EX-05 board	x 1
5. Uni-polar Stepper motor 12V 100Ω 7.5 degree/step	x 1
6. IDC-10 cable	x 1

Procedure

Input data for 1-Phase stepper motor driving has 2 groups. One are 1, 2, 4 and 8 for anti-clockwise direction. Another are 8, 4, 2 and 1 for clockwise direction. See the driving data in the table P-1

1.1 Make the **Comand1** button and write the experiment code as :

```
Private Sub Command1_Click()
   Lefts = False
   Rights = True
   Do
       DoEvents
       Out &H378, 1
          Call delay
       Out &H378, 2
          Call delay
       Out &H378, 4
          Call delay
       Out &H378, 8
          Call delay
   Loop Until Lefts = True
End Sub
```

Step	Phase-4	Phase-3	Phase-2	Phase-1
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	1	0	0	0

(A) Step data in Left rotation

Step	Phase-4	Phase-3	Phase-2	Phase-1
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

(B) Step data in Right rotation

Table P1 Sequnce operation of stepper motor's coil in 1-phase full step driving

1.2 Interface P-Board with EX-05 board by IDC-10 cable via DATA BUS conector and connect the stepper motor to EX-05 board. Must check the correct phase before connection.

1.3 Apply the power supply to P-Board and EX-05 board.

1.4 Run the experiment program from step 1.1

1.5 Click **Command1**, program will be send values 1,2,4 and 8 in order to. Each data will be separated by delay routine. In same time, **`Do....Loop Until`** command will check the **Command2** status. Observe motor operation.

1.6 About Delay routine, have many techniques to make it. One is using timebase timer. The advantage of this technique is delay time still equal in all computer. But this techniques has limit of speed. It cause speed of program may be not enouh to run some condition. The minimum value is 0.01 second. Sample program in VB6 can show below :

```
Sub delay()
   Times = Timer
   Do
      DoEvents
   Loop Until Timer >= Times + 0.01
End Sub
```

This routine will verify value in **Timer** *variable. This value is internal computer timebase (***Times***) plus 0.01 in Second unit. If internal time is equal or more than* **Times +0.01**, *program will be out of delay routine loop.* **`Doevents`** *command is assist to run another routine such as can click* **Command2** *button in delay loop.*

1.6 Another delay routine can show in this step below. This code can increase the speed working but depend on each computer.

```
Sub delay()
   For i = 1 To HScroll1.Value
      DoEvents
   Next i
End Sub
```

This routine use **`For...Next`** *command and* **Hscroll** *slide bar for adjust speed. In case the loop has long time period and protect the program hang, must insert* **`Doevents`** *command in the loop.*

1.7 For the Right rotation, must make the **Command2** button. See the code below :

```
Private Sub Command2_Click()
   Lefts = True
   Rights = False
   Do
      DoEvents
      Out &H378, 8
         Call delay
      Out &H378, 4
         Call delay
      Out &H378, 2
         Call delay
      Out &H378, 1
         Call delay
   Loop Until Rights = True
End Sub
```

Overall operation will be similar Left rotaion. Different is only rotate data.In this direction use value 8,4,2 and 1 instead. The window program can see in figure P1.

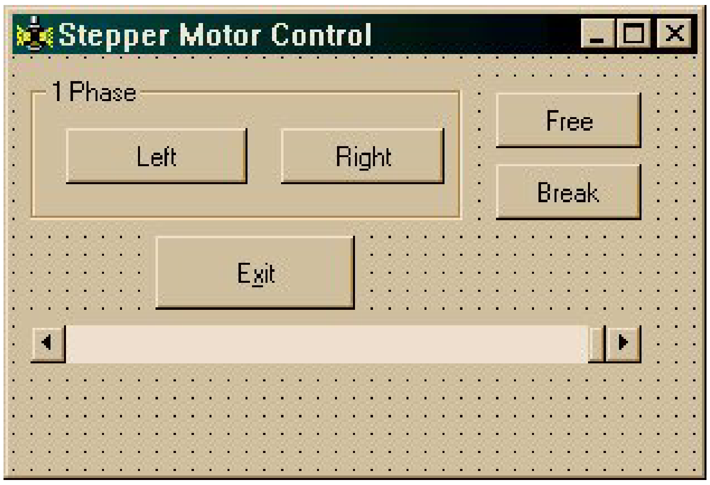


Figure-P1 Stepper motor control program window

About sample experiment program for Parallel port can see in Lab05.VBP in Parallel_port/Lab/Lab05 folder and Lab06.VBP file in Serial_port/Lab/Lab06 folder All files are contain in INEXís Computer Interface CD-ROM.

Sample experiment-2

2-Phase Stepper motor driving

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-05 board	x 1

3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

4. +12V 2A external DC power supply fo EX-05 board	x 1
5. Uni-polar Stepper motor 12V 100Ω 7.5 degree/step	x 1
6. IDC-10 cable	x 1

Procedure

2.1 The 2-phase stepper motor drivng data format is shown in table P2 . Tha data value will be 9,3,6 and 12 (in decimal) for Left rotation and 12, 6, 3 and 9 for Right rotation. Edit the Visual BASIC code in **Command1** and **Command2** button from the simple experiment-1 as :

```
Dim i As Integer
Dim Lefts, Rights As Boolean
Private Sub Command1_Click()
   Lefts = False
   Rights = True
   Do
       DoEvents
       Out &H378, 9
          Call delay
       Out &H378, 3
          Call delay
       Out &H378, 6
          Call delay
       Out &H378, 12
          Call delay
   Loop Until Lefts = True
End Sub
```

Step	Phase-4	Phase-3	Phase-2	Phase-1
1	1	0	0	1
2	0	0	1	1
3	0	1	1	0
4	1	1	0	0

(A) Step data in Left rotation

Step	Phase-4	Phase-3	Phase-2	Phase-1
1	1	1	0	0
2	0	1	1	0
3	0	0	1	1
4	1	0	0	1

(B) Step data in Right rotation

Table P2 Sequnce operation of stepper motor's coil in 2-phase full step driving

```
Private Sub Command2_Click()
   Lefts = True
   Rights = False
   Do
       DoEvents
       Out &H378, 12
          Call delay
       Out &H378, 6
          Call delay
       Out &H378, 3
          Call delay
       Out &H378, 9
          Call delay
   Loop Until Rights = True
End Sub
```

2.2 Interface P-Board with EX-05 board by IDC-10 cable via DATA BUS conector and connect the stepper motor to EX-05 board. Must check the correct phase before connection.

2.3 Apply the power supply to P-Board and EX-05 board.

2.4 Run the experiment program from step 2.1

2.5 Observe the stepper motor operation and compare with the experiment-1 (step 1.4)

The 2-phase full step technique can drive more torque but require more power 2 times.

About sample experiment program for Parallel port can see in Lab06.VBP in Parallel_port/Lab/Lab06 folder and Lab06.VBP file in Serial_port/Lab/Lab06 folder All files are contain in INEXís Computer Interface CD-ROM.

Sample experiment-3

Half-step mode

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-05 board	x 1

3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

4. +12V 2A external DC power supply fo EX-05 board	x 1
5. Uni-polar Stepper motor 12V 100Ω 7.5 degree/step	x 1
6. IDC-10 cable	x 1

Procedure

The half-step technique cause stepper motor driving more resolution 2 times than full step mode. Then it need data 8 value for cycle. See the sequence operation in Table P3. The data for Left rotation are 9, 1, 3, 2, 6, 4, 12 and 8. The Right rotation data are 8, 12, 4, 6, 2, 3, 1 and 9

3.1 Write the experiment code for Command1 and Commande2 button as :

```
Private Sub Command1_Click()
   Lefts = True
   Rights = False
   Do
      DoEvents
      Out &H378, 9
         Call delay
      Out &H378, 1
         Call delay
      Out &H378, 3
         Call delay
      Out &H378, 2
         Call delay
      Out &H378, 6
         Call delay
      Out &H378, 4
         Call delay
      Out &H378, 12
         Call delay
      Out &H378, 8
         Call delay
      Out &H378, 0
         Call delay
   Loop Until Rights = True
End Sub
```

Step	Phase-4	Phase-3	Phase-2	Phase-1
1	1	0	0	1
2	0	0	0	1
3	0	0	1	1
4	0	0	1	0
5	0	1	1	0
6	-	1	0	0
7	1	1	0	0
8	1	0	0	0

(A) Step data in Left rotation

Step	Phase-4	Phase-3	Phase-2	Phase-1
1	1	0	0	0
2	1	1	0	0
3	0	1	0	0
4	0	1	1	0
5	0	0	1	0
6	0	0	1	1
7	0	0	0	1
8	1	0	0	1

(B) Step data in Right rotation

Table P3 Sequnce operation of stepper motor’s coil in half-step mode

```
Private Sub Command2_Click()
   Lefts = False
   Rights = True
   Do
      DoEvents
      Out &H378, 8
         Call delay
      Out &H378, 12
         Call delay
      Out &H378, 4
         Call delay
      Out &H378, 6
         Call delay
      Out &H378, 2
         Call delay
      Out &H378, 3
         Call delay
      Out &H378, 1
         Call delay
      Out &H378, 9
         Call delay
      Out &H378, 0
         Call delay
   Loop Until Lefts = True
End Sub
```

About sample experiment program for Parallel port can see in Lab07.VBP in Parallel_port/Lab/Lab07 folder and Lab06.VBP file in Serial_port/Lab/Lab06 folder All files are contain in INEXís Computer Interface CD-ROM.

3.2 Interface P-Board with EX-05 board by IDC-10 cable via DATA BUS conector and connect the stepper motor to EX-05 board. Must check the correct phase before connection.

3.3 Apply the power supply to P-Board and EX-05 board.

3.4 Run the experiment program from step 3.1

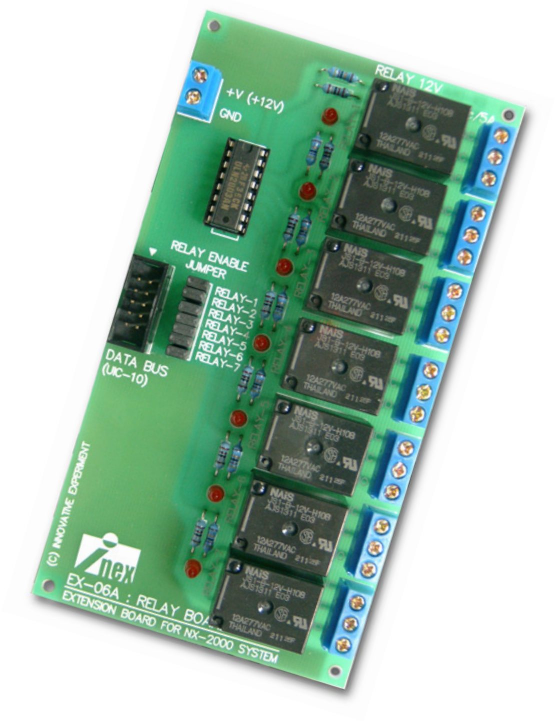
3.5 Observe the stepper motor operation and compare the result with the experiment-1 and 2

Stepper motor will rotate slower than experiment-1 and 2 but it rotate more resolution 2 times.

Experimenters can see detail and get the sample code to run from EX-07.VBP file in Computer interfaceís CD-ROM

EX-06A
7-ch. Relay driver board

(C) Innovative Experiment Co.,Ltd.



Features :

- Drive 12V relay 7 channels with LED status
- Require +12V external power supply
- Selection by jumper settings
- Connect to P-Board, S-Board and U-Board with DATA BUS connector (10-pin IDC header connector)

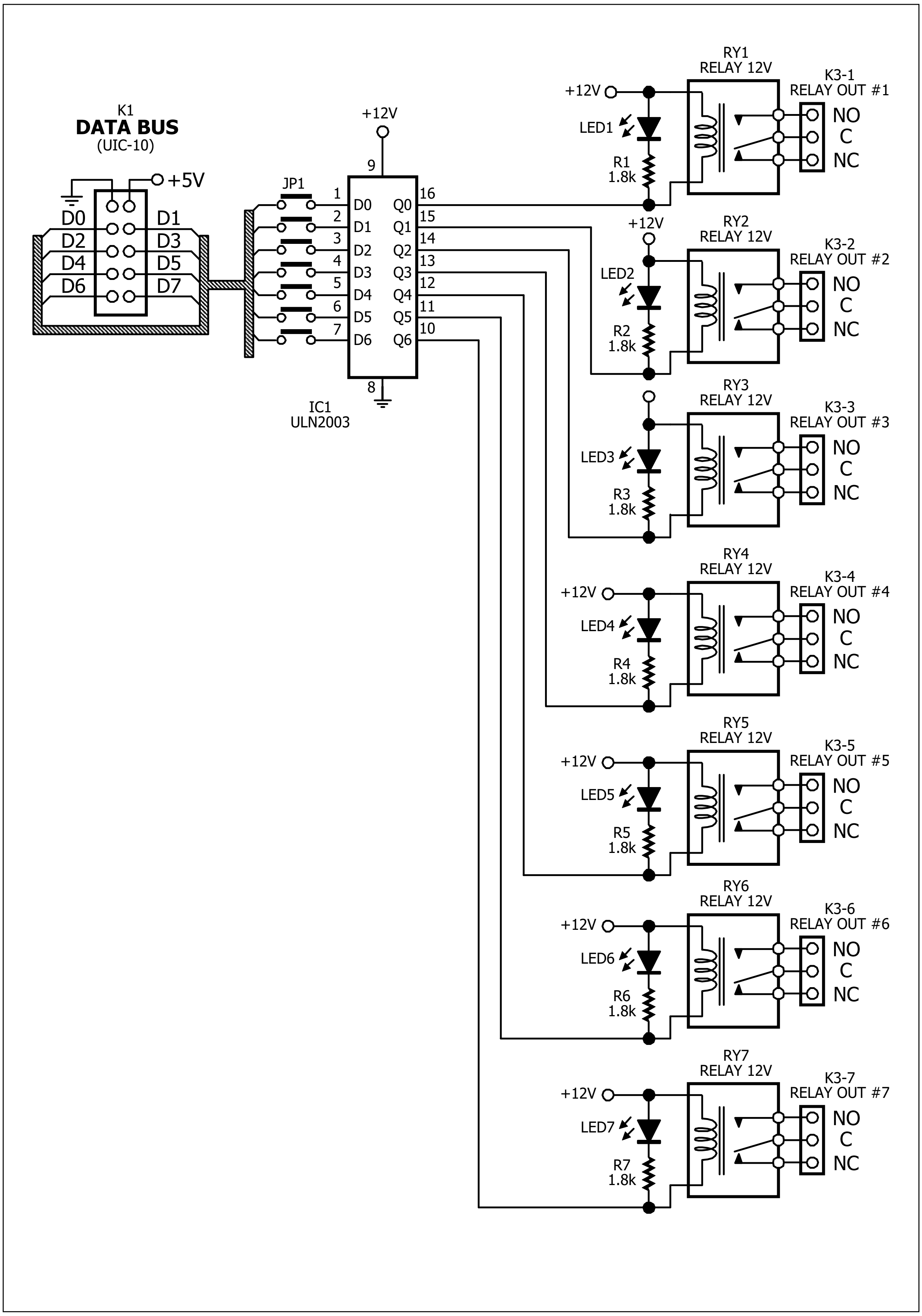


Figure-1 EX-06A Relay driver board schematic

Circuit Description

EX-06A board provides 7 mechanical relays. Ther are drived by ULN2003 driver. The schematic diagram seein figure 1.

The heart of this board is ULN2003 IC. It has 7 inverter driver. All input are connected to JP1 to JP8 jumper. Experimenters can select by manual. Another end of jumpers are connected DATA BUS connector. Prepare for connect with P-Board, S-Board and U-Board.

The logic to activate is High. ULN2003 will be invert to Low at output. Current flow pass the coil. It causethe relay coil active. Relay contact will switch from NC to NO. LED at each relay will light.

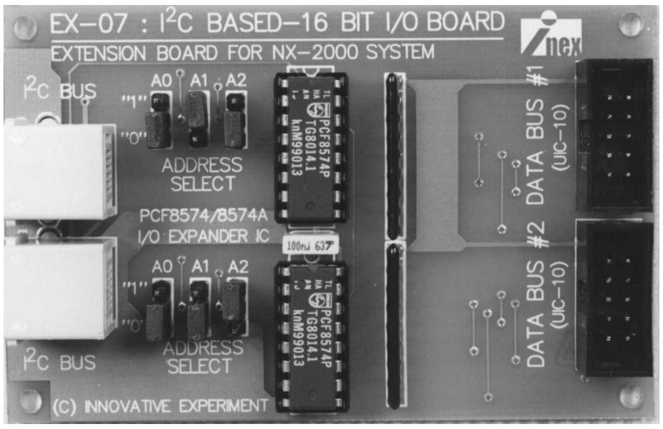
In case interface with Parallel port, can interface with Data port or Control port. If connected with Data port, can control up to 7 channels from D0 to D6. If connected with Control port, can control only 4 channels and must becareful about invert logic in some pin.

INNOVATIVE EXPERIMENT

EX-07

I^{2}C-based I/O Expansion board

(C) Innovative Experiment Co.,Ltd.



EX-07 : I2C BASED-16 BIT I/O BOARD
EXTENSION BOARD FOR NX-2000 SYSTEM
I2C BUS
ADDRESS SELECT
PCF8574/8574A I/O EXPANDER IC
DATA BUS #1 (UIC-10)
DATA BUS #2 (UIC-10)
(C) INNOVATIVE EXPERIMENT

Features :

- Connect I^{2}C BUS via modular jack
- Expanable 16-bit/board. up to 4 boards, 64-bit I/O
- Select address by jumper
- Available ìDATA BUSî connector for P-board, S-Board and U-Board

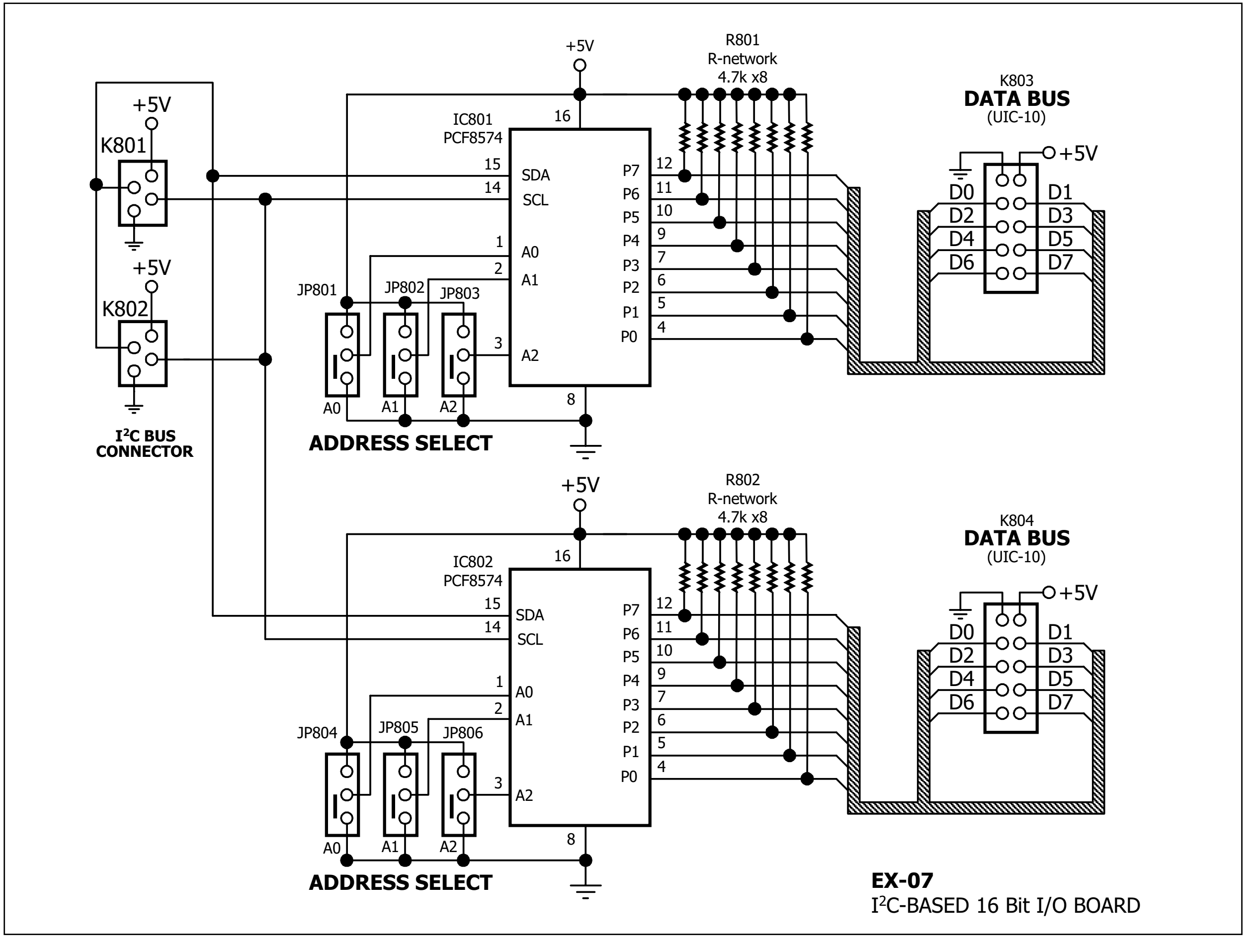


Figure-1 EX-07 I^{2}C-based I/O Expansion board schematic

Circuit description

The figure 1 shows EX-07 board schematic. The heart of this board is PCF8574A, 8-bit input/output (I/O) expander for the two-line bidirectional bus (I2C) IC. The PCF8574A provides general-purpose remote I/O expansion for most microcontroller families via the I2C interface (serial clock (SCL), serial data (SDA)). The device features an 8-bit quasi-bidirectional I/O port P0ñP7), including latched outputs with high-current drive capability for directly driving LEDs. Each quasi-bidirectional I/O can be used as an input or output without the use of a data-direction control signal. At power on, the I/Os are high. In this mode, only a current source to Vcc is active. An additional strong pullup to Vcc allows fast rising edges into heavily loaded outputs. This device turns on when an output is written high and is switched off by the negative edge of SCL. The I/Os should be high before being used as inputs.

EX-07 board has two PCF8574A. Then one board can support 16 I/O and can expand 4 boards 64 I/O. One board must select two different address by jumpers. One PCF8574A must set one address by A0-A2 jumpers.

I2C bus of EX-07 board use modular jack 6P4C pin type. Experimeters can conect direct to P-Board and S-Board.

I^{2}C Interface

I^{2}C communication with this device is initiated by a master sending a start condition, a high-to-low transition on the SDA I/O while the SCL input is high. After the start condition, the device address byte is sent, most-significant bit (MSB) first, including the data direction bit ($R/\overline{W}$). This device does not respond to the general call address. After receiving the valid address byte, this device responds with an acknowledge, a low on the SDA I/O during the high of the acknowledge-related clock pulse. The address inputs (A0ñA2) of the slave device must not be changed between the start and the stop conditions.

The data byte follows the address acknowledge. If the $R/\overline{W}$ bit is high, the data from this device are the values read from the P port. If the $R/\overline{W}$ bit is low, the data are from the master, to be output to the P port. The data byte is followed by an acknowledge sent from this device. If other data bytes are sent from the master, following the acknowledge, they are ignored by this device. Data are output only if complete bytes are received and acknowledged. The output data will be valid at time, after the low-to-high transition of SCL and during the clock cycle for the acknowledge.

A stop condition, a low-to-high transition on the SDA I/O while the SCL input is high, is sent by the master.

Sample experiment

Expansion I/O port of Parallel port via I²C bus

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-07 board	x 1
3. EX-01 board	x 1
4. EX-03 board	x 1

5. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

6. I2C bus cable	x 1
7. IDC-10 cable	x 1

Procedure

Interface EX-07 board programming

1. Put jumpers to select address of PCF8574A #1 on EX-07 board to 000. Another one select 001 address.

2. Connect P-Board with EX-07 board by I²C cable and connect EX-01 board with IDC-10 cable following in figure P1.

3. Write the code to send data to output of PCF8574A for driving LED on EX-01 board as :

```
Private Sub Command1_Click()
Sendout (&HAA)
End Sub
Private Sub Sendout(B As Integer)
   Call I2CStart                    'Start
   Call Send8BIT(&H70)              'Send Address Word
   Call Ack                         'Acknowledge
   Call Send8BIT(B)                 'Send Data
   Call Ack                         'Acknowledge
   Call I2CStop                     'Stop
End Sub
Private Sub I2CStart()
   Out &H37A,Inp(&H37A) Or 1        'SDA=1
   Out &H37A,Inp(&H37A) Or 2        'SCL=1
   Out &H37A,Inp(&H37A) And &HFE    'SDA=0
   Out &H37A,Inp(&H37A) And &HFD    'SCL=0
End Sub
```

```
Private Sub I2CStop()
   Out &H37A,Inp(&H37A) And &HFE     'SDA=0
   Out &H37A,Inp(&H37A) Or 2         'SCL=1
   Out &H37A,Inp(&H37A) Or 1         'SDA=1
End Sub
Private Sub Send1()
   Out &H37A,Inp(&H37A) Or 1         'SDA=1
   Out &H37A,Inp(&H37A) Or 2         'SCL=1
   Out &H37A,Inp(&H37A) And &HFD     'SCL=0
End Sub
Private Sub send0()
   Out &H37A,Inp(&H37A) And &HFE     'SDA=0
   Out &H37A,Inp(&H37A) Or 2         'SCL=1
   Out &H37A,Inp(&H37A) And &HFD     'SCL=0
End Sub
Private Sub Ack()
   Out &H37A,Inp(&H37A) Or 1         'SDA=1
   Out &H37A,Inp(&H37A) Or 2         'SCL=1
   Out &H37A,Inp(&H37A) And &HFD     'SCL=0
End Sub
```

The interface programís screen can show in figure P2.

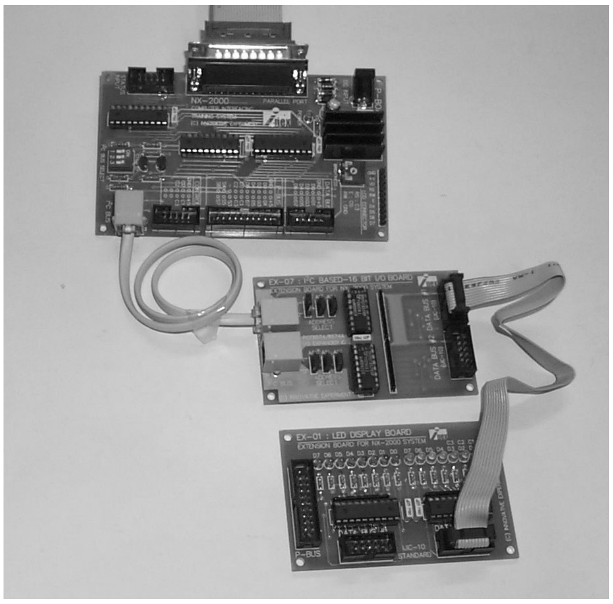


Figure P1 Shows interfacing of P-Board, EX-07 and EX-01 board

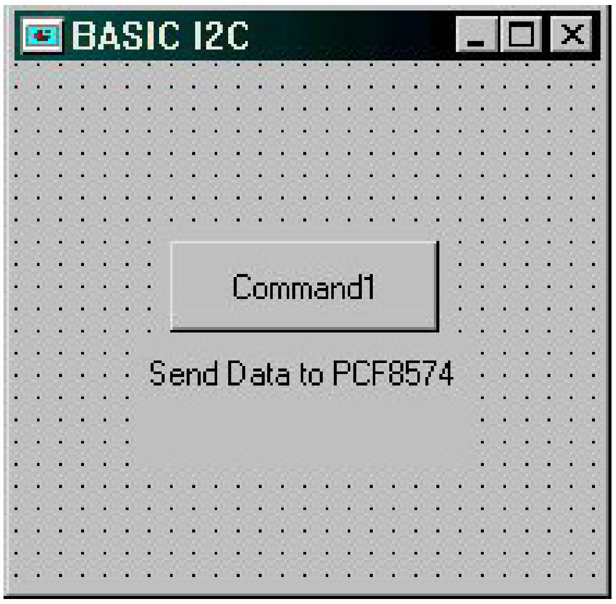


Figure P2 Shows the program's screen of sending data to output ofPCF8574A.

About sample experiment program for Parallel port can see in Lab11A.VBP in Parallel_port/Lab/Lab11 folder and Lab09.VBP file in Serial_port/Lab/Lab09 folder All files are contain in INEXís Computer Interface CD-ROM.

Reading input from I²C bus device

Before read data from I²C bus, the first thing to do is set the bus to idle. The last bit (LSB) of address must set to ë1í. It means host request to read data from device.

In VB program writing for reading data 8 bit will be use **DAT** function to loops 8 times for reading data and compare the LSB bit of data. In using **DAT** function, it will return 8-bit data. The sample program can show below :

```
Private Function DAT()
  For I = 7 To 0 Step -1
    Out &H37A, Inp(&H37A) Or 1                    'SDA=1
    Out &H37A, Inp(&H37A) Or 2                    'SCL=1
      If (Inp(&H379) And &H80) = &H80 Then        'Read SDA
         DAT1 = 2 ^ I Or DAT1
      End If
    Out &H37A, Inp(&H37A) And &HFD                'SCL=0
  Next I
  DAT = DAT1           'Data 8 Bit
End Function
```

Reading input from EX-07 board

4. Except use EX-07 board for sending the output data, experimenters can use EX-07 to reading data. In this experiment will use EX-03 for input source. Add routine below to read the switch data by **Timer1** :

```
Private Sub Timer1_Timer()
   Call I2CStart              'Start
   Call Send8BIT(&H71)        'Send Control Word
   Call Ack                   'Acknowledge
      Text1.Text = Hex$(DAT)
   Call Ack
   Call I2CStop
End Sub
```

Reading data step are :

(1) Send START condition.

(2) Send PCF8574A address and define to read mode.

(3) Send Acknowledge condition.

(4) Read data by **DAT** function.

(5) Send Acknowledge following STOP condition.

From this code, use **Timer** in operation. It help to read data continuous but must set the **Interval** value to suitable. In this code set **Interval** value = 100. HEX$ command is in front of **DAT** function use for setting data to Hexadecimal.

5. Interfac P-Board, EX-07, EX-01 and EX-03 following figure P1.

6. Run the experiment code in step 4. The screen of this program is shown in figure P3.

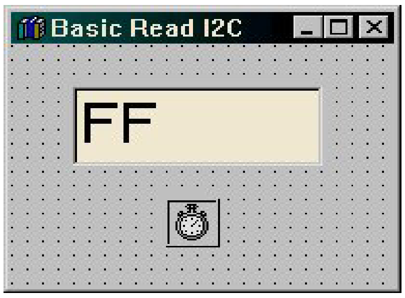


Figure P3 I2C bus reading data program screen

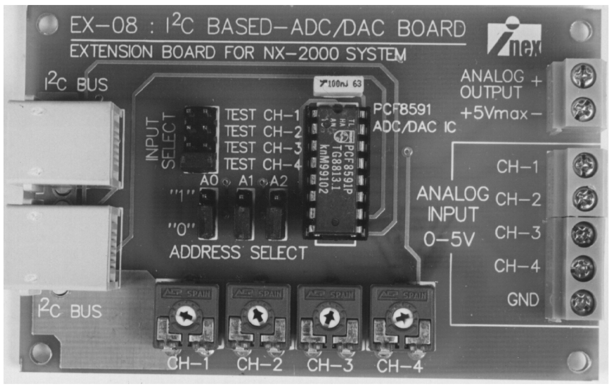
7. Select data input at push-button swithces on EX-03 board. Observe the result on programís screen at **Text1**.

About sample experiment program for Parallel port can see in Lab11B.VBP in Parallel_port/Lab/Lab11 folder and Lab09.VBP file in Serial_port/Lab/Lab09 folder All files are contain in INEXís Computer Interface CD-ROM.

EX-08

I²C-based ADC/DAC board

(C) Innovative Experiment Co.,Ltd.



Features :

- Connect I²C BUS via modular jack
- 8-bit A/D converter 4 ch. Input 0 to +5V.
- 8-bit D/A converter 1 channel
- Expanable 4 analog inputs and 1 analog output per board. Up to 8 boards, 32 analog inputs and 8 analog outputs.
- Select address by jumper

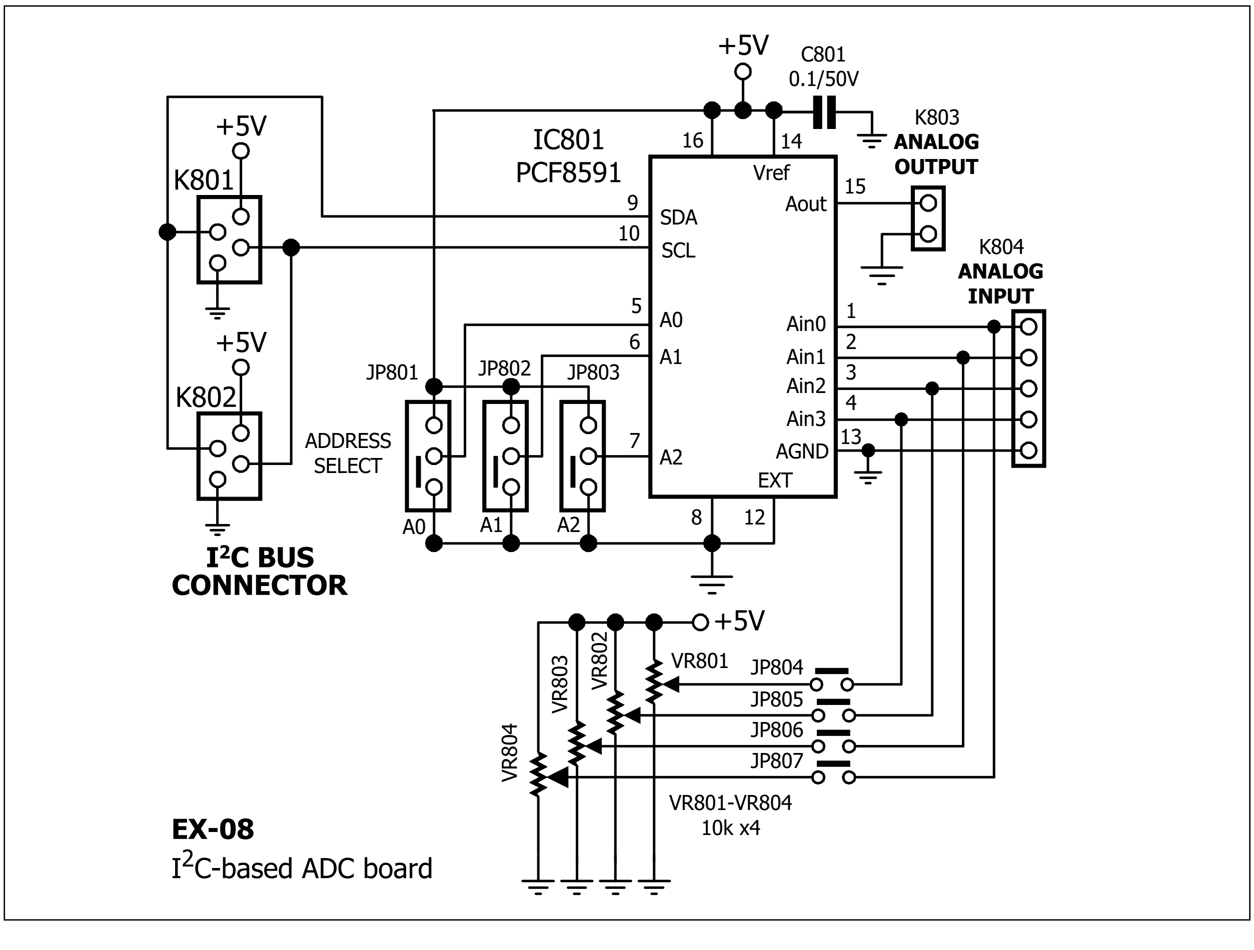


Figure-1 EX-08 I²C-based ADC/DAC board schematic

Circuit description

The figure 1 is EX-08 schematic. The main device is PCF8591 IC. The PCF8591 is a single-chip, single-supply low power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I2C-bus interface.

Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I2C-bus without additional hardware. Address, control and data to and from the device are transferred serially via the two-line bidirectional I2C-bus.

The functions of the device include analog input multiplexing, on-chip track and hold function, 8-bit analog-to-digital conversion and an 8-bit digital-to-analog conversion. The maximum conversion rate is given by the maximum speed of the I2C-bus.

Normally, if use one EX-08 board, suggess to set its address to 000. Interface with P-Board and S-Board, the EX-08 board use I2C bus modular jack same EX-07 board and can expand to 8 boards if set different address.

The input of EX-08 board can select 2 sources. One is from external via terminal blocks. Another is variable resistors on EX-08 board. Experimenters can select by jumpers JP704-JP707.

The analog output of EX-08 has only one channel per board. Output voltage range is 0 to +5V selected by software.

Sample experiment

Analog interface with Parallel port via I^{2}C bus

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-08 board	x 1

3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

4. 0-5V DC power supply 4 outputs or 4 of single output 0-5V DC power supply.

5. Digital multimeter with probe	x 1
6. I2C bus cable	x 1

Procedure

Reading analog input from EX-08 continuous

Step interfacing of PCF8591 are :

(1) Send START condition.

(2) Send PCF8591 address; 000 (A0 to A2 connect to ground) and define to write mode (bit LSB clear to ë0í : R/W = 0).

(3) Wait for ACK condition from PCF8591.

(4) Send control data to PCF8591; 45H. It means Enable analog output, Set analog input to single mode, Reading continuous and Start to read ADC channel 1.

(5) Wait for ACK status from PCF8591.

(6) Send STOP condition.

(7) Send START condition.

(8) Send PCF8591 address; 000 again but set the LSB to ë1í (R/W = 1) for beginning read data from analog input.

(9) Wait for ACK status from PCF8591.

(10) Read input of ADC channel 1.

(11) Send MAck (Master Ack) condition to PCF8591

(12) Read input of ADC channel 2.

(13) Send MAck (Master Ack) condition to PCF8591

(14) Read input of ADC channel 3.

(15) Send MAck (Master Ack) condition to PCF8591

(16) Read input of ADC channel 4.

(17) Wait for ACK condition from PCF8591.

(18) Send STOP condition

From all step above, can convert to VB code below :

```
' Read 4 analog input in continuous
Private Sub Timer1_Timer()
   Call I2CStart
   Call Send8BIT(&H90)
   Call Ack
   Call Send8BIT(&H45)
   Call Ack
   Call I2CStop
   Call I2CStart
   Call Send8BIT(&H91)
   Call Ack
   Text1.Text = (DAT * 5) / 255
   Call MAck
   Text2.Text = (DAT * 5) / 256
   Call MAck
   Text3.Text = (DAT * 5) / 256
   Call MAck
   Text4.Text = (DAT * 5) / 256
   Call Ack
   Call I2CStop
End Sub
```

`MAck` sub-routine is used to send Acknowledge condition from computer to PCF8591. The detail of this routine can show below :

```
' MAck subroutine
Private Sub MAck()
   Out &H37A, Inp(&H37A) And &HFE    'SDA=0
   Out &H37A, Inp(&H37A) Or 2        'SCL=1
   Out &H37A, Inp(&H37A) And &HFD    'SCL=0
   Out &H37A, Inp(&H37A) Or 1        'SDA=1
End Sub
```

About **I2CStart**, **Send8BIT**, **Ack** *and* **I2CStop** *sub-routine can see detail in EX-07 documentation.*

Sending MAck condition to slave will opposite Ack condition. In the code set 4 Textboxes for containing ADC data of 4-analog channels ; **Text1**, **Text2**,**Text3** and **Text4**. Value in **TextBox** is voltage reading from PCF8591 inputs because they are results from calculating 8-bit digital data. The screen of this program is shown in figure P1.

About sample experiment program for Parallel port can see in Lab12A.VBP in Parallel_port/Lab/Lab12 folder and Lab1001.VBP file in Serial_port/Lab/Lab10 folder. All files are contain in INEXís Computer Interface CD-ROM.

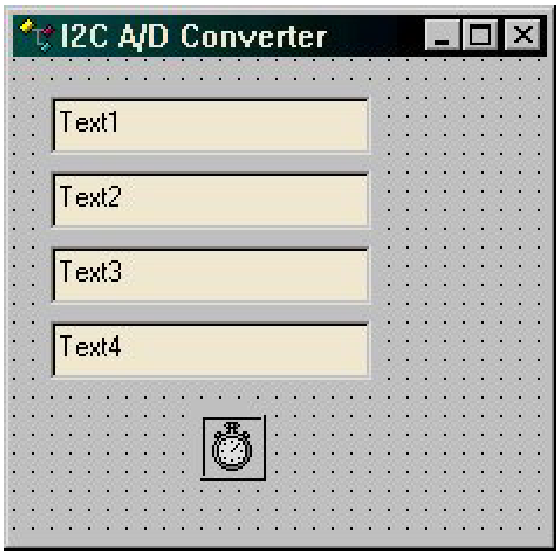


Figure P1 Shows the screen of EX-13A.VBP file that test the operation of EX-08 board.

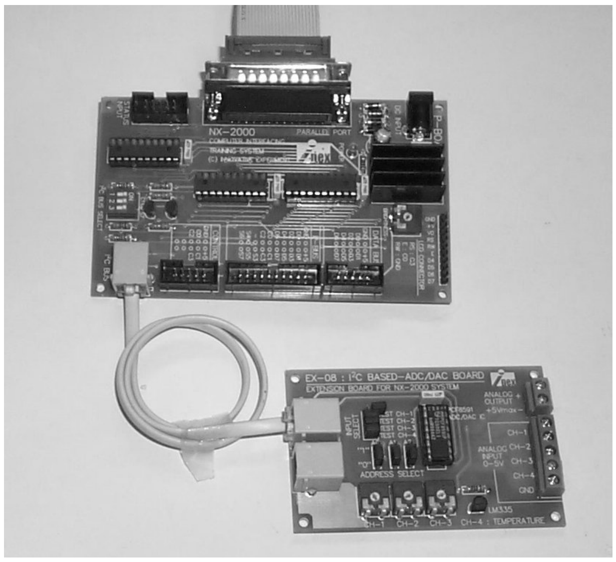


Figure P2 Shows interface P-Board with EX-08

(1) Fit jumper to select PCF8591 address on EX-08 board to 000

(2) Connect P-Board with EX-08 board by I^{2}C cable following in figure P2.

(3) Open Lab12A.VBP file in Parallel_port/Lab/Lab12 folder in INEXís Computer Interface CD-ROM and run it.

(4) Fit jumper to select TEST CH-1 analog input (select analog source on-board)

(5) Adjust variable resistor at CH-1 following figure P3

(6) Observe the vlatage changing in **Text1** box.

(7) Change to test with another input. Test folloing step 4 and 5. See operation as :

Text2 *box for CH-2*, **Text3** *box for CH-3 and* **Text4** *box for CH-4*

(8) Fit jumper at TEST position of all channel.

(9) Adjust the variable resisitor in each channel and observe the operation at all Textbox.on the screen.

Calculation digital data to analog

In 8-bit ADC, experimenters can calculate the resolution as :

Resolution = *Full scale voltage / 8-bit Digital value maximum in decimal*

= 5/256 = 0.0195V or 20mV approximation

For voltage input calculation (back from reading) is :

Input voltage = (Reading data X Full scale voltage) / 8-bit Digital value maximum

Example : ADC data from PCF8591 is 128, Input voltage is (128x5)/256 = 2.5V

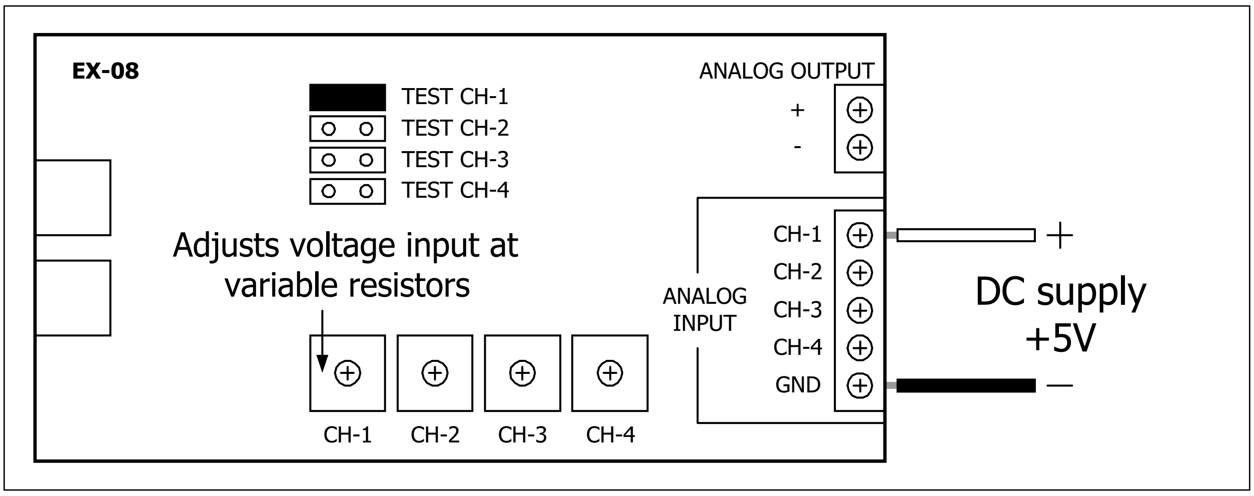


Figure P3 Preparation of EX-08 is in experiment step 4 to 7

Writing data to DAC module in PCF8591

PCF8591 has a 8-bit Digital to Analog Converter (DAC) module. The step of send ing data to DAC module are :

(1) Send START condition.

(2) Send PCF8591 address and define to write mode (R/W bit is ì0î).

(3) Wait for ACK condition back from PCF8591.

(4) Send control data; 44H to PCF8591 for enable the output analog.

(5) Wait for ACK condition back from PCF8591.

(6) Send data 0-255 to analog output.

(7) Wait for ACK condition back from PCF8591.

(8) Send STOP condition.

From all step above, can convert to VB code as :

```
' Sending data to DAC module in PCF8591 routien
Private Sub Text5_Change()
   If Val(Text5.Text) > 5 Then Text5.Text = 5
   Call I2CStart
   Call Send8BIT(&H90)
   Call Ack
   Call Send8BIT(&H44)
   Call Ack
   Call Send8BIT(Val(Text5.Text) * 51.2)
   Call Ack
   Call I2CStop
End Sub
```

*This routine use a TextBox; **Text5** to define data that send to PCF8591. This value will multiply by 51.2 (51.2 number come from8-bit value in deciamal; 256 devided by the full sclae voltage ; 5)*

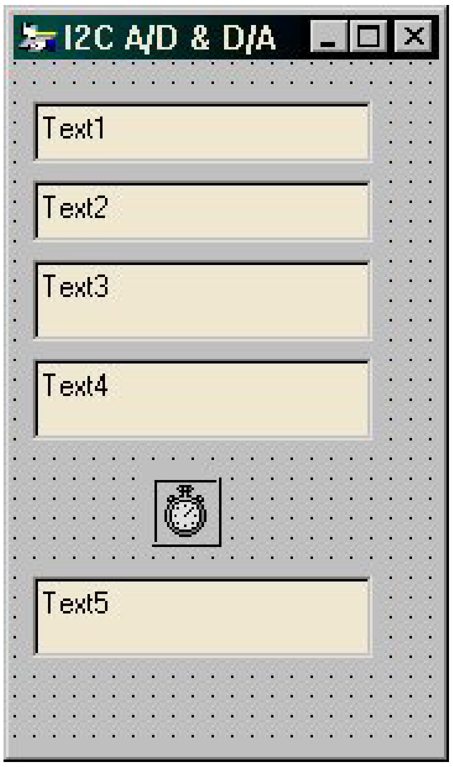


Figure P4 Shows the program's screen of Lab12B.VBP file. It is ADC/DAC demonstration program of PCF8591

(10) Fit jumpers to select address 000 for EX-08 board.

(11) Connect P-Board with EX-08 board by I^{2}C cable following figure P2

(12) Open Visual BASIC and open Lab12B.VBP to run. The program screen is shown in figure P4

(13) Put valtage value into Text5 box. Limit is 5.

(14) Use a digital meter set to DC volt. Measure voltage at ANALOG OUTPUT terminal on EX-08 board following figure P5 and compare with value in Text5 box.

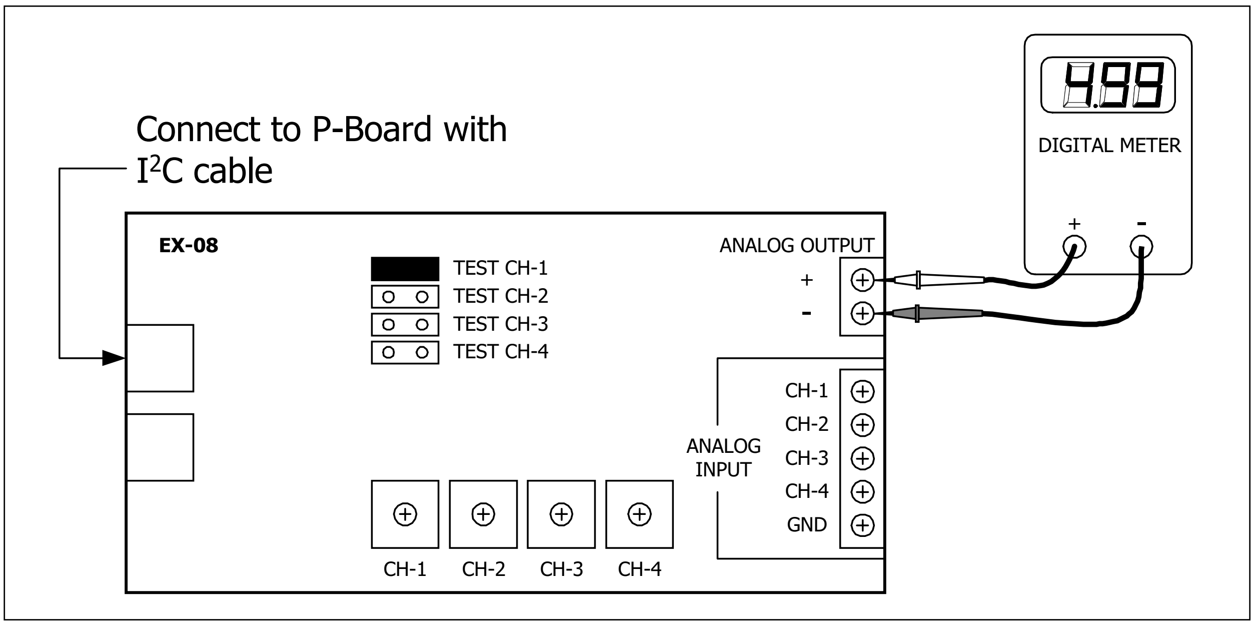


Figure P5 Shows interface diagram for experiment step 14

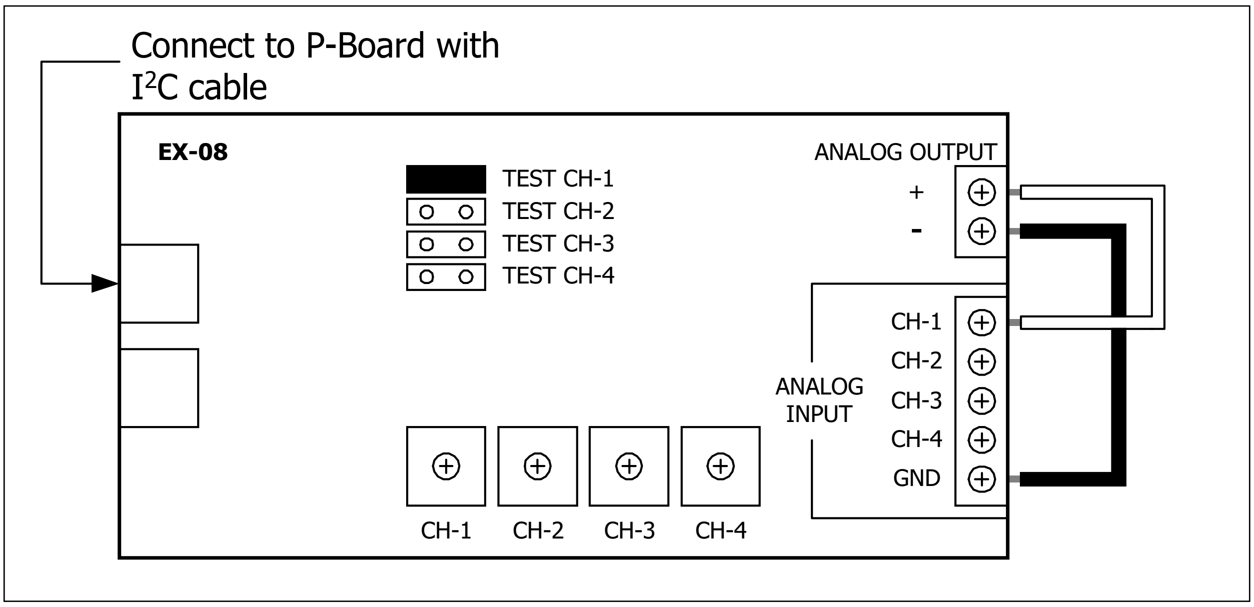


Figure P6 Shows interface diagram for experiment step 15

(15) Connect ANALOG OUTPUT back to CH-1 input on EX-08 board following figure P6.

(16) Put voltage value in Text5 box again and not over 5.

(17) Observe the changing in Text1 box. Does it relate with value in Text5 box ?

*In step 13 and 14 are testing the conversion performance of PCF8591. Value in **Text5** box will be converted to digital data after that send to PCF8591 to convert as analog voltage and send to ANALOG OUTPUT*

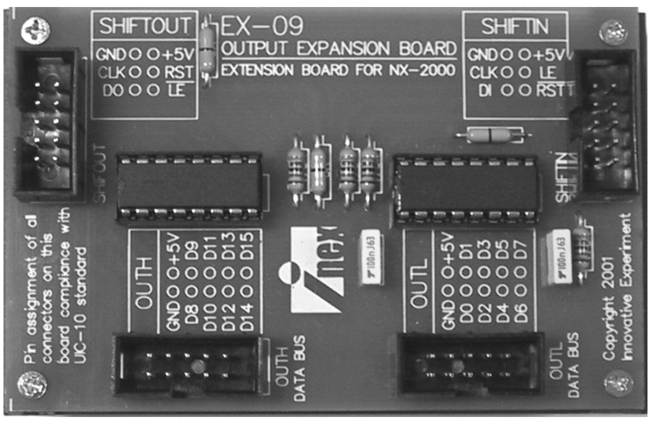
*In step 15 to 17 are experiment of using ADC and DAC module within PCF8591 to gether. Program will convert data in **Text5** box to digital data and send to DAC module in PCF8591 converts to analog voltage. After that this voltage is sent back to analog input of ADC module in PCF8591 for convert to digital data and shows in **Text1** box.*

About sample experiment program for Parallel port can see in Lab12B.VBP in Parallel_port/Lab/Lab12 folder and Lab1001.VBP file in Serial_port/Lab/Lab10 folder All files are contain in INEXís Computer Interface CD-ROM.

EX-09

Synchronous Output Expansion Board

(C) Innovative Experiment Co.,Ltd.


SHIFTOUT
GND +5V
CLK RST
DO LE
EX-09
OUTPUT EXPANSION BOARD
EXTENSION BOARD FOR NX-2000
SHIFTIN
GND +5V
CLK LE
DI RST
OUTH
GND +5V D8 D9 D10 D11 D12 D13 D14 D15
OUTL
GND +5V D0 D1 D2 D3 D4 D5 D6 D7
OUTH DATA BUS
OUTL DATA BUS
Pin assignment of all connectors on this board compliance with UIC-10 standard
Copyright 2001 Innovative Experiment

Features :

- Expand output port 8 bits 2 groups per board
- Use the shift register IC in operation
- Connection to other boards is unlimit (but recommendation of 4 boards per one of P-Board/S-Board or U-Board)
- Connect with P-Board and S-Board via DATA BUS
- Connect with U-Board via SHIFT OUT connector

Basic concept

The figure 1 shows the port expansion basic diagram by synchronous shift register technique. It has only one data line (DI). The transfer method must use serial transfer. It use main clock signal (CLK) for controlling data transfer to relate between the transmitter (here is computer) and receiver (here is shift register IC). Addition 2 control signals are Reset (RST) and Latch Enable (LE) for holding data to output.

The shift register IC for this technique will use SIPO (Serial In Parallel Out). Start by clear or reset data to ì0î. After that shift data one bit into DI and follow with one clock. Data will shift into the shift register. If do not sending LE signal, output data still not change. Next send data until complete 8 bits and following send LE signal to latch output data. All data within shift register will transfer to output, see the timing diagram in figure 2.

If the shift register has a serial data output, user can expand the I/O more by connect SO (serial data output : SO) to Data input pin (DI) of next shift register IC. THe port will be expanded from 8 to 16 bit, 24 bit, 32 bit or etc.

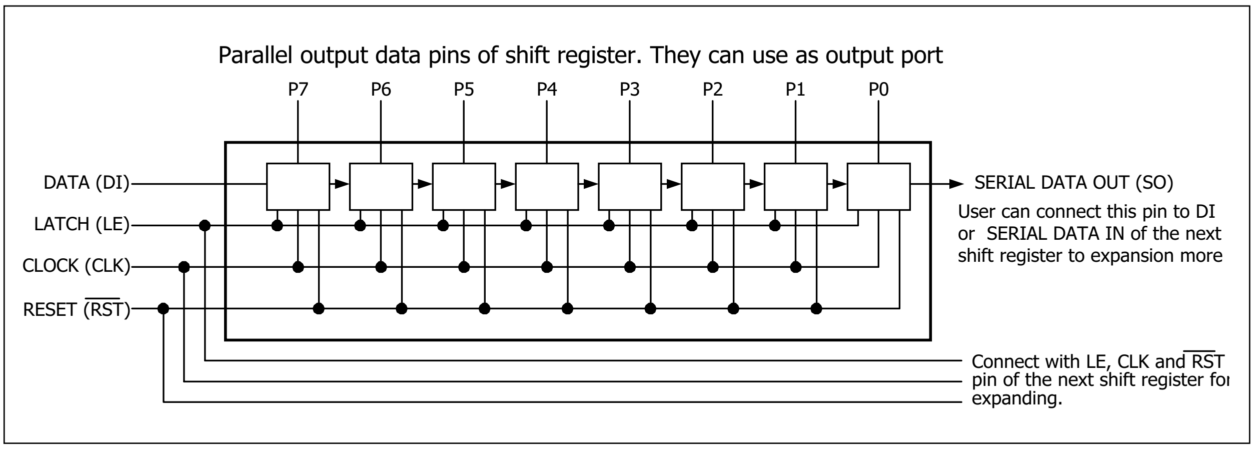

Parallel output data pins of shift register. They can use as output port
P7
P6
P5
P4
P3
P2
P1
P0
DATA (DI)
LATCH (LE)
CLOCK (CLK)
RESET $\overline{(RST)}$
SERIAL DATA OUT (SO)
User can connect this pin to DI or SERIAL DATA IN of the next shift register to expansion more
Connect with LE, CLK and $\overline{RST}$ pin of the next shift register for expanding.

Figure 1 Basic diagram of output port expansion by synchronous shift register technique.

The limitation of this technique are :

(1) Overall speed will reduce. If there is many data bits, the shifting time of shift register will be increase.

(2) Current supply will reduce. If there is many port, it means many device in same system. The currrent comsumption will be increase. The system cannot supply more device in long term.

(3) More noise. If there is more expansion, the signal cable must have long length more. It cause the noise can access to interfere the system.

Circuit description

Figure 3 shows the full schematic diagram of EX-09 board ; Synchronous output port expansion board. This board can expand the output port up to 16 bit. per board and can connect daisy chain to unlimit (however user must check the hardware limitation).

The heart of this board is 74HC595 shift register IC SISO/SIPO type. 74HC595 has 8-bit shift register. This circuit use 2 of 74HC595. It cause EX-09 expand the output port to 16 bit.

Serial data and control signal will send pass SHIFT IN connector. This connector pin assignment are INEXís UIC-10. Then user can connect direct to DATA BUS of P-Board and S-Board. CLK is connected to D0 pin, LE - D1, DI - D2 and RST is connected to D3. Data signal send to 74HC595 for expanding to 8-bit output port. All signal are connected to OUTL connector. Addition, the serial data output from SO pin is sent to SI pin of the second 74HC595 to expand more 8 bits. These data are connected to OUTH connector. Thus, EX-09 baord can expand output port to 16 bits from 4 signal lines.

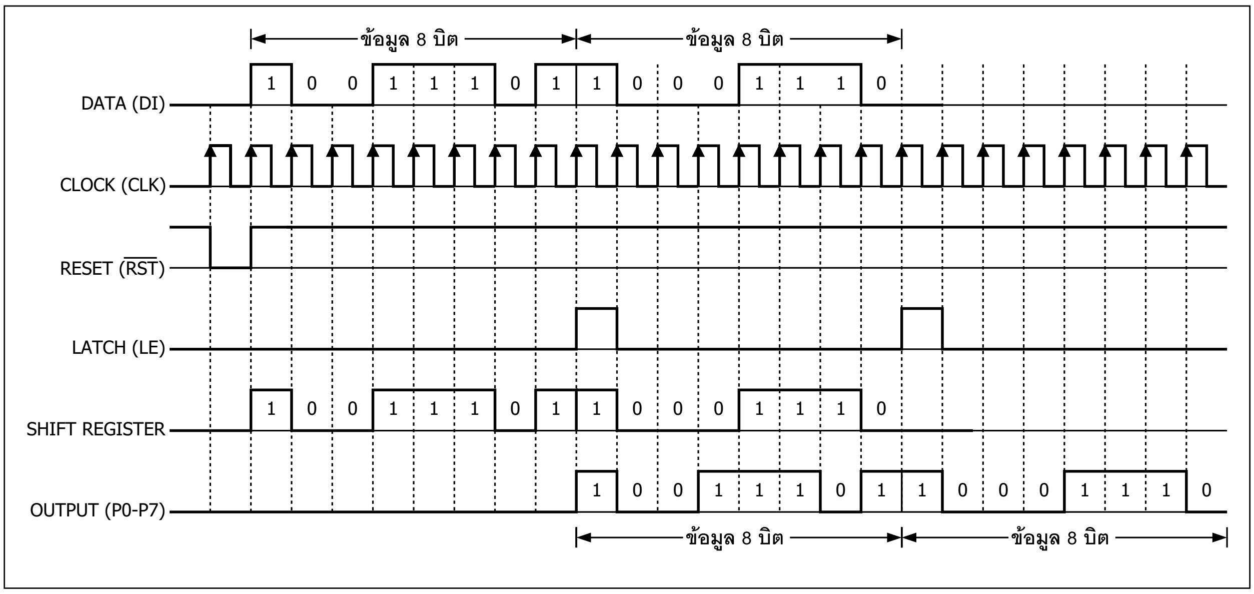


Figure 2 Timing diagram of shift register operation for port expansion

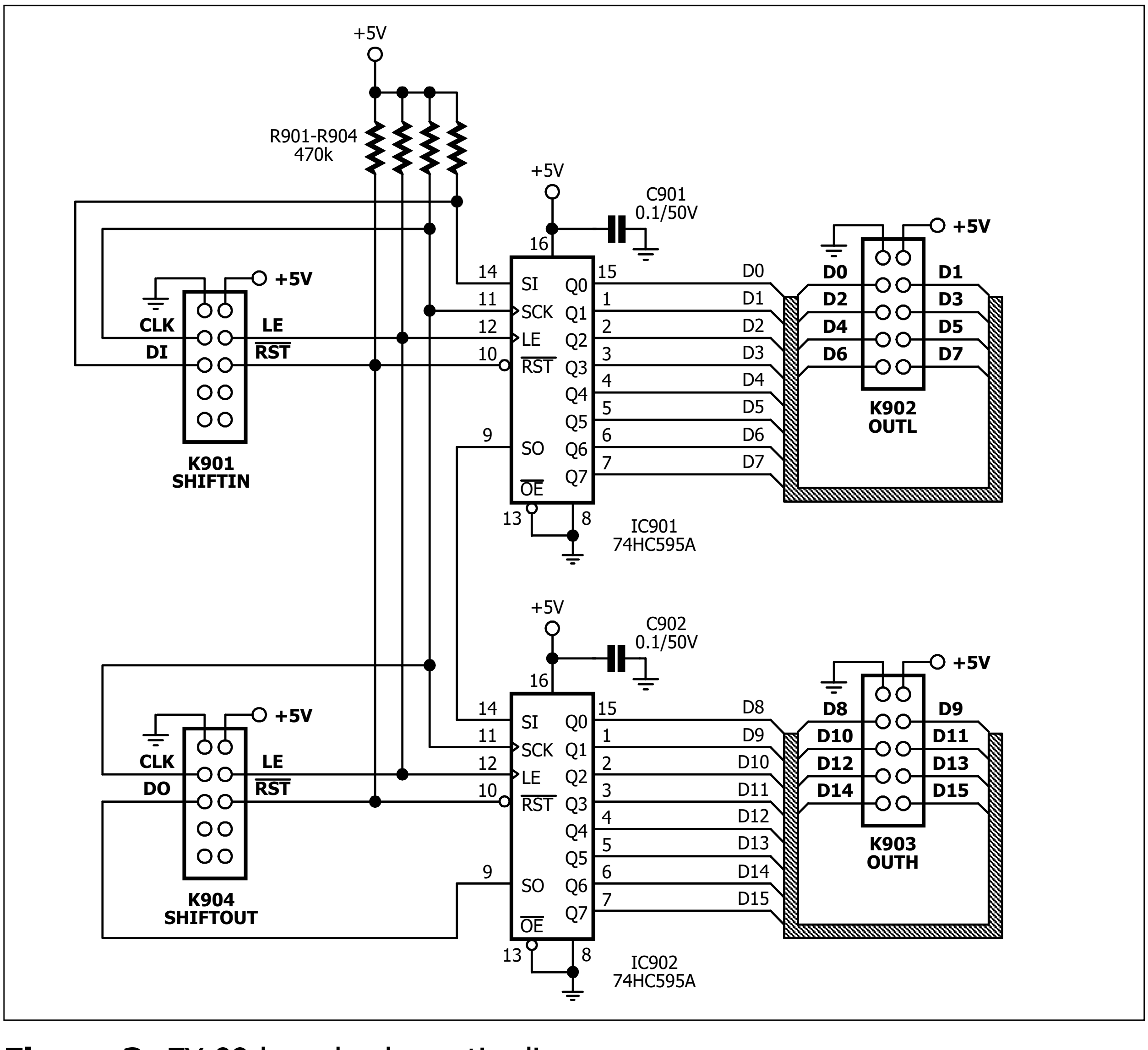


Figure 3 EX-09 board schematic diagram

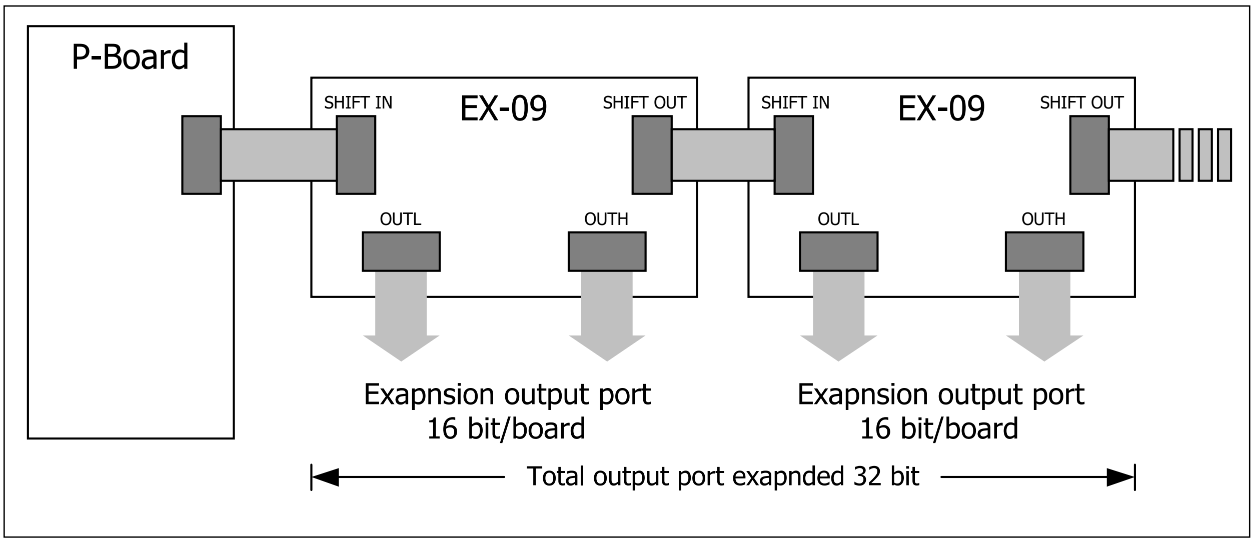


Figure 4 Show the daisy chain connection of EX-09 to expand the output port

In fact, user can expand more. By connect LE, CLK and RST signal from SHIFT OUT connector to SHIFT IN connector of the next EX-09 board. If connect 2 boards, the expansion are 32 bits from same 4 signal lines. Recommended maximum expansion is 4 boards for a P-Board or S-Board. Figure 4 shows the expansion port of EX-09.

Sample experiment -1
Expansion output port for parallel port

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-09 board	x 1
3. EX-01 board	x 1

4. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

5. IDC-10 cable	x 3

Procedure

1.1 Connect IDC-10 cable from DATA BUS connector of P-Board to SHIFT IN connector on EX-09 board and connect IDC-10 cable from OUTH and OUTL connector from EX-09 to DATABUS1 and 2 of EX-01 board

1.2 Open Visual BASIC and place control on form and change the name following the figure P1-1 . Write the code below :

```
Private Sub InitialShiftIn()
    Out &H378, &H8
End Sub
Private Sub CLK()
    Out &H378, Inp(&H378) Xor &H1
    Out &H378, Inp(&H378) Xor &H1
End Sub
Private Sub RST()
    Out &H378, Inp(&H378) Xor &H8
    Out &H378, Inp(&H378) Xor &H8
End Sub
Private Sub LE()
    Out &H378, Inp(&H378) Xor &H2
    Out &H378, Inp(&H378) Xor &H2
End Sub
Private Sub DI(ByVal Logic As Boolean)
    If Logic Then
        Out &H378, Inp(&H378) Or &H4
    Else
        Out &H378, Inp(&H378) And &HFB
    End If
End Sub
```

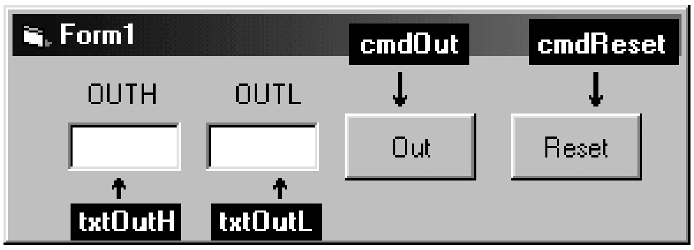


Figure P1-1 Shows VB form of the LAB13A.VBP file for output port expansion experiment by using EX-09 board

1.3 Write the sub-routine code of sending 8-bit data

```
Private Sub Shift8Bit(ByVal Data As Byte)
Dim i As Integer
    For i = 7 To 0 Step -1
        If (Data And (2 ^ i)) = (2 ^ i) Then
            DI True
        Else
            DI False
        End If
        CLK
    Next i
End Sub
```

1.4 Write the code for **Form_Load** , **cmdReset_Click** and **cmdOut_Click** event as :

```
Private Sub Form_Load()
    InitialShiftIn
    RST
    LE
End Sub

Private Sub cmdReset_Click()
    RST
    LE
End Sub

Private Sub cmdOut_Click()
    Shift8Bit CByte("&H" & txtOutH.Text)
    Shift8Bit CByte("&H" & txtOutL.Text)
    LE
End Sub
```

*From the experiment code, program received data from **txtOutH** and convert to byte data and send to Shift8Bit sub-routine for shiftinh data to 74HC595 and clock signal After that, bring the data from **txtOutL** to convert to byte data and send too. After shfit 16bit complete, will send latch signal. All data from EX-09 board will out to LED on EX-01 board. At OUTH will be the **txtOutH** data and OUTL will be the **txtOutL** data.*

About sample experiment program for Parallel port can see in Lab13A.VBP in Parallel_port/Lab/Lab13 folder and Lab11.VBP file in Serial_port/Lab/Lab11 folder. All files are contain in INEXís Computer Interface CD-ROM.

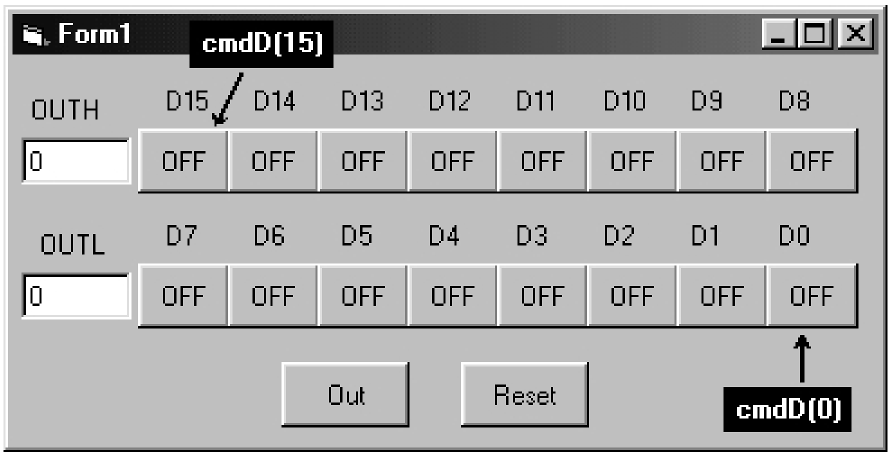


Figure P1-2 Shows VB form of the LAB13B.VBP file for output port expansion experiment by using EX-09 board and Array technique.

1.5 Add control and arrange form following the figure P1-2 Define all button as Array control in same name but different **Index** from 0 to 15.

1.6 Write the code for cmdD_Click(Index As Integer) event as :

```
Private Sub cmdD_Click(Index As Integer)
Dim i As Integer
Dim tmp As Byte
    If cmdD(Index).Caption = "OFF" Then
        cmdD(Index).Caption = "ON"
    Else
        cmdD(Index).Caption = "OFF"
    End If

    tmp = 0
    For i = 0 To 7
        If cmdD(i + 8).Caption = "ON" Then tmp = tmp + (2 ^ i)
    Next i
    txtOutH.Text = Hex(tmp)
    Shift8Bit tmp

    tmp = 0
    For i = 0 To 7
        If cmdD(i).Caption = "ON" Then tmp = tmp + (2 ^ i)
    Next i
    txtOutL.Text = Hex(tmp)
    Shift8Bit tmp
    LE
End Sub
```

About sample experiment program for Parallel port can see in Lab13B.VBP in Parallel_port/Lab/Lab13 folder. All files are contain in INEXís Computer Interface CD-ROM.

*Click any **cmdD(Index)** button, program will verify about Caption of that button as ON or OFF and invert Caption to opposite. After that verify Caption of each button at OUTH group for reading each bit then convert to Hex format data and send out to SHIFT IN port of EX-09 board.*

When complete 8-bit, program will work with OUTL data group same OUTH. After complete all 16-bit, program will send latch signal to EX-09 board for get the latest data output to OUTx connector. All data will appeare on EX-01 board.

Sample experiment -2

Drive stepper motor via EX-09

Material :

1. P-Board Parallel port Interface board	x 1
2. EX-09 board	x 1
3. EX-05 board with stepper motor	x 1

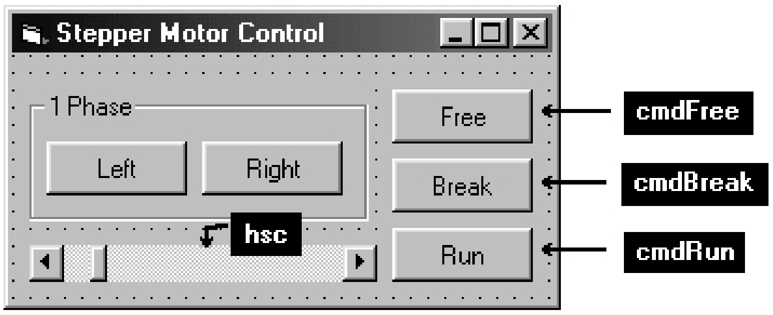
4. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

5. External power supply for stepper motor	x 1
6. IDC-10 cable	x 3

Procedure

2.1 Connect IDC-10 cable from OUTLconnector of EX-09 to DATA BUS connector of EX-05 board. Connect stepper motor and power supply to EX-05 board.

2.2 Make thee control program. Place the control, define name and set properties following the figure P2-1



Name	hsc
Max	4
Min	1
LargeChange	5
SmallChange	1
Value	100

Figure P2-1 Shows the screen of Stepepr motor control experiment program and detail of each button and control.

2.3 This program will work in the loop of **cmdRun_click** event for control the motor. Looping variable is boolean type and declare the variable at General Declare as :

```
Dim flgRun As Boolean
Dim flgLeft As Boolean, flgRight As Boolean, flgFree As Boolean, flgBreak As
Boolean
```

Make the routine for changing the motor driving 4 events : **cmdFree_Click**, **cmdBreak_Click**, **cmdLeft_Click** and **cmdRight_Click**

```
Private Sub cmdBreak_Click()
    flgLeft = False
    flgRight = False
    flgBreak = True
    flgFree = False
End Sub
Private Sub cmdFree_Click()
    flgLeft = False
    flgRight = False
    flgBreak = False
    flgFree = True
End Sub
Private Sub cmdLeft_Click()
    flgLeft = True
    flgRight = False
    flgBreak = False
    flgFree = False
End Sub
Private Sub cmdRight_Click()
    flgLeft = False
    flgRight = True
    flgBreak = False
    flgFree = False
End Sub
```

2.4 Make the delay routine. The delay time can define at hsc variable join with 16-bit shifting. The detail of this code as :

```
Private Sub Delay()
Dim a As Single
    a = Timer
    Do While Timer < a + (hsc.Value / 100)
        DoEvents
    Loop
End Sub
Private Sub Shift16Bit(Optional ByVal OutH As Byte = 0, Optional ByVal OutL As
Byte = 0)
    Shift8Bit OutH
    Shift8Bit OutL
    LE
End Sub
```

2.5 Write the code for **cmdRun_Click** and **Form__Unload** event as :

```
Private Sub cmdRun_Click()
    If flgRun Then
        flgRun = False
        cmdRun.Caption = “Run”
        Exit Sub
    Else
        flgRun = True
        cmdRun.Caption = “Stop”
    End If
    Do While flgRun
        If flgLeft Then
            Shift16Bit 0, 8
            Delay
            Shift16Bit 0, 4
            Delay
            Shift16Bit 0, 2
            Delay
            Shift16Bit 0, 1
            Delay
        ElseIf flgRight Then
            Shift16Bit 0, 1
            Delay
            Shift16Bit 0, 2
            Delay
            Shift16Bit 0, 4
            Delay
            Shift16Bit 0, 8
            Delay
        ElseIf flgBreak Then
            Shift16Bit 0, 3
            Delay
        ElseIf flgFree Then
            Shift16Bit 0, 0
            Delay
        End If
    Loop
End Sub
```

```
Private Sub Form_Unload(Cancel As Integer)
    Shift16Bit 0, 0                 ‘ Release Motor
    End
End Sub
```

2.6 Run the experiment program. Click Run button, motor still not work. Because all motor control variable is set default to False nothing send out. Then add some code below into **Form_Load** event to set the start direction.

```
flgLeft = True
```

About sample experiment program for Parallel port can see in Lab14A.VBP in Parallel_port/Lab/Lab14 folder. All files are contain in INEXís Computer Interface CD-ROM.

Drive stepper motor with LED via EX-09

2.7 Add control and Array control following the figure P2-2. Define **cmdD** button has **Index** value from 8 to 15.

2.8 Declare variable for contain port data; OUTH and OUTL at General Declaration

```
Dim OutH As Byte, OutL As Byte
```

2.9 Edit the code of **cmdRun_Click** event to send status of OUTH port instead 0 value and collect OUTL port status after send data to drive stepper motor. This code below is example code :

```
Shift16Bit OutH, 1
OutL = 1
Delay
Shift16Bit OutH, 2
OutL = 2
Delay  ....
```

2.10 Write the code for **cmdD_Click** and **cmdReset_Click** event as :

```
Private Sub cmdD_Click(Index As Integer)
Dim i As Integer
Dim tmp As Byte
    If cmdD(Index).Caption = "OFF" Then
        cmdD(Index).Caption = "ON"
    Else
        cmdD(Index).Caption = "OFF"
    End If
    tmp = 0
    For i = 0 To 7
        If cmdD(i + 8).Caption = "ON" Then tmp = tmp + (2 ^ i)
    Next i
    txtOutH.Text = Hex(tmp)
    OutH = tmp
    Shift16Bit OutH, OutL
End Sub
Private Sub cmdReset_Click()
Dim i As Byte
    For i = 8 To 15
        cmdD(i).Caption = "OFF"
    Next i
    txtOutH.Text = 0
    OutH = 0
    Shift16Bit OutH, OutL
End Sub
```

About sample experiment program for Parallel port can see in Lab14B.VBP in Parallel_port/Lab/Lab14 folder. All files are contain in INEXís Computer Interface CD-ROM.

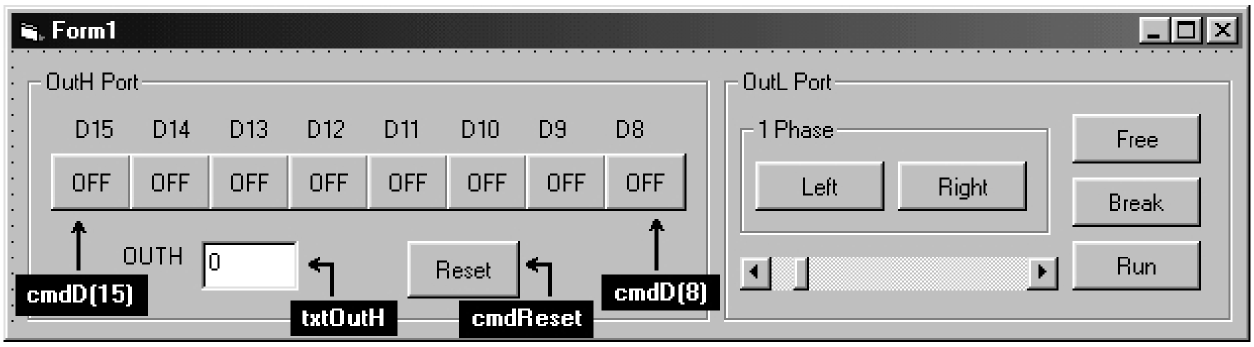


Figure P2-2 Shows the screen of Stepper motor control and LED display experiment program

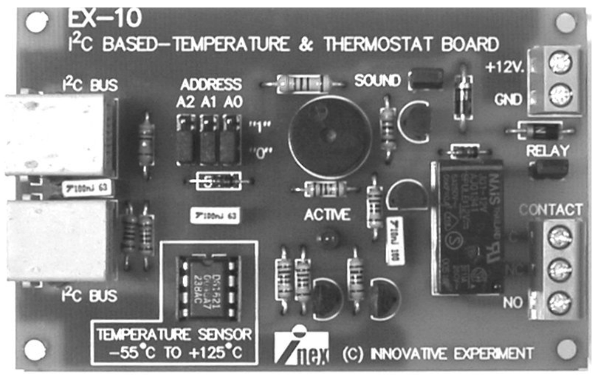
2.11 Run the program. Click Run button to strart the motor and turn-on or off LED.

*The program operation is using **OUTH** and **OUTL** variable to store port status everytime to change. In sending data, will send all status data of both variable together. Status at both port will correct always.*

EX-10

I^2C bus-based temperature board

(C) Innovative Experiment Co.,Ltd.



Features :

● Use DS1621 ICs Temperature range is -55^{0}C to +125^{0}C. For EX-10 baord can operation in 0 to 50 ^{0}C

● Connect I^2C BUS via modular jack for P-Board and S-Board

● Thermostat based output, selectable High temp/Low temp trigger point

● Output device in Thermostat mode include LED, Piezo speaker abd Relay. Selected by jumper.

● Expandability up to 8 boards. Selection of address by jumper settings

Circuit description

Figure 1 shows EX-10 board schematic. Main component is DS1621 Digital Thermometer and Thermostat. DS1621 provides 9ñbit temperature readings which indicate the temperature of the device. The thermal alarm output, TOUT, is active when the temperature of the device exceeds a userñdefined temperature TH. The output remains active until the temperature drops below user defined temperature TL, allowing for any hysteresis necessary.

In this schematic, SDA and SCL pin of DS1621 iare connected to K1001 and K1002, I2C bus modular jack for interface with P-Board and S-Board.Jumper JP1001-JP1003 are used to select address of EX-10 board. Then EX-10 board can connect together total 8 boards maximum.

Thermal alarm output of DS1620 will connect with 2 output devices ; Piezo speaker for sounding and Relay for activated switch. User can select another one or both by jumper JP1004 and JP1005. One of thermal alarm device is LED indicator; LED1001. It will light if temperature value reach to TH point

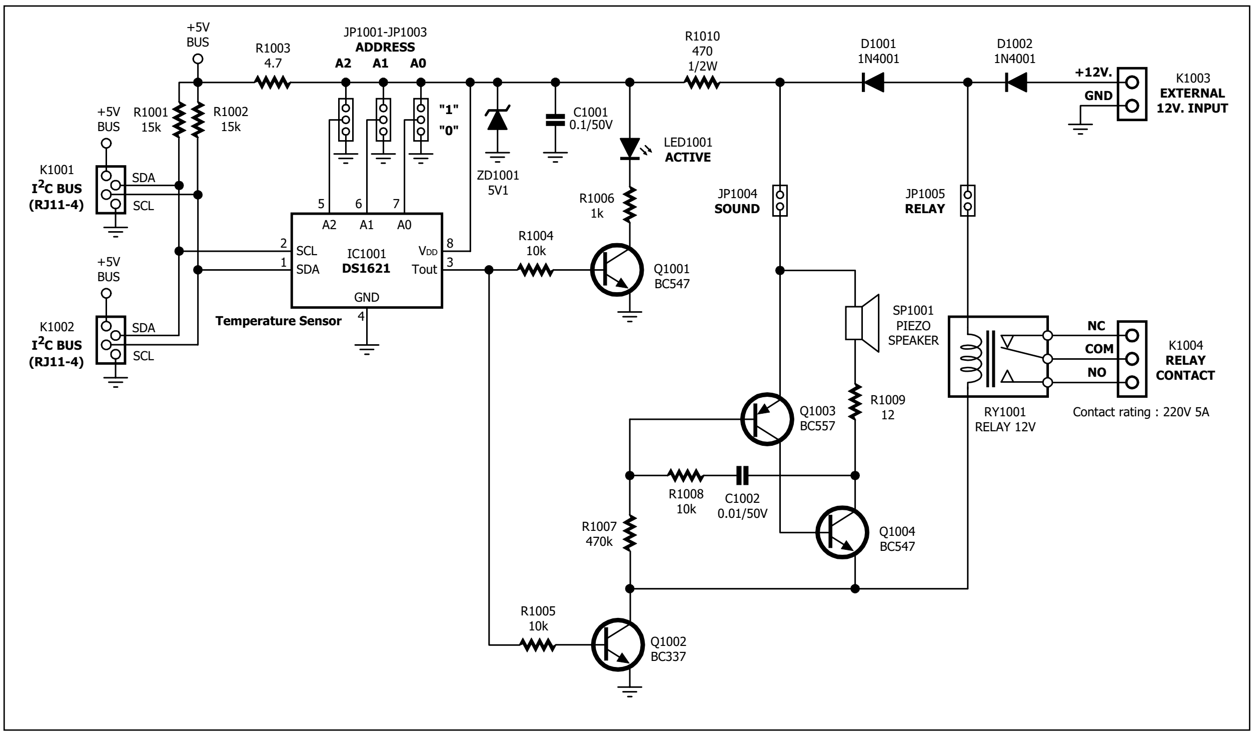


Figure 1 Schematic diagram of EX-10 Temperature & Thermostat board

Relay; RY1001 provides a 3-pins termnal block for connect external load. The contact rating is 220VAC 5A If user would like to use relay, must apply +12V supply to EX-10 board supply terminal block instead using supply from I2C bus jack. In case not using relay, user can use supply voltage +5V from I2C bus jack and must remove jumper JP1005.

However all output device will operate coorect when work in thermostat mode only. Thermal output pin will out logoc ë1í when the temperature value reach to TH point. In program setting, must write the program to set POL bit of Configuration and Status register of DS1621.

User can see more info about DS1621 from www.maxim-ic.com. However in INEXís Computer interface CD-ROM contains all info about DS1621 in DATASHEET folder. Must see it for evaluate about register and control command.

Master mode (Computer)	**DS1621 mode**	**Condition**	**Description**
TX	RX	START	Bus Master initiates a START condition.
TX	RX	<ADDRESS,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	ACH	Bus Master sends Access Config command protocol.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	02H	continuous conversion.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	START	Bus Master generates a repeated START condition.
TX	RX	<address,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	A1H	Bus Master sends Access TH command.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	28H	^{o}C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	00H	^{o}C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	START	Bus Master generates a repeated START condition.
TX	RX	<address,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	A2H	Bus Master sends Access TL command.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	0AH	^{o}C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	00H	^{o}C.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	START	Bus Master generates a repeated START condition.
TX	RX	<address,0>	Bus Master sends DS1621 address; R/W = 0.
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	EEH	
RX	TX	ACK	DS1621 generates acknowledge bit.
TX	RX	STOP	Bus Master initiates STOP condition.

Table 1 The example of DS1621 function table (refer data in table, set TH to +40^{o}C and TL +10^{o}C)

Sample experiment DS1621 versus Parallel port

Material :

1. P-Board Parallel port Interface board x 1

2. EX-10 board x 1

3. Computer that available a parallel port, install Windows (98SE/ME/XP) and Visual BASIC V5.0 or higher

4. I2C bus cable x 1

Procedure

Reading temperature value from DS1621

Before reading the temperature value from Temperature register in DS1621, must define DS1621 to **start convert** following all steps below :

(1) Send START condition.

(2) Send DS1621ís address on EX-10 board and clear LSB bit as ì0î to write mode.

(3) Wait for ACK condition from DS1621

(4) Send command &HEE to start conversion.

(5) Wait for ACK condition from DS1621.

(6) Send STOP condition

Controlling DS1621 stop convert has steps as :

(1) Send START condition.

(2) Send DS1621ís address on EX-10 board and clear LSB bit as ì0î to write mode.

(3) Wait for ACK condition from DS1621.

(4) Send command &H22 to stop conversion.

(5) Wait for ACK condition from DS1621.

(6) Send STOP condition

Reading Temperature register has steps as :

(1) Send START condition.

(2) Send DS1621ís address on EX-10 board and clear LSB bit as ì0î to write mode.

(3) Wait for ACK condition from DS1621.

(4) Send command &HAA to read value from DS1621ís Temperature register.

(5) Wait for ACK condition from DS1621.

(6) Send START condition.

(7) Send DS1621ís address on EX-10 board and clear LSB bit as ì1î to read mode.

(8) Wait for ACK condition from DS1621.

(9) Read MSB bit of Temperature register.

(10) Send Master ACK condition to DS1621.

(11) Read LSB bit of Temperature register.

(12) Send Master Not ACK condition to DS1621.

(13) Send STOP condition.

1. Build form and change control names following the firgure P1-1

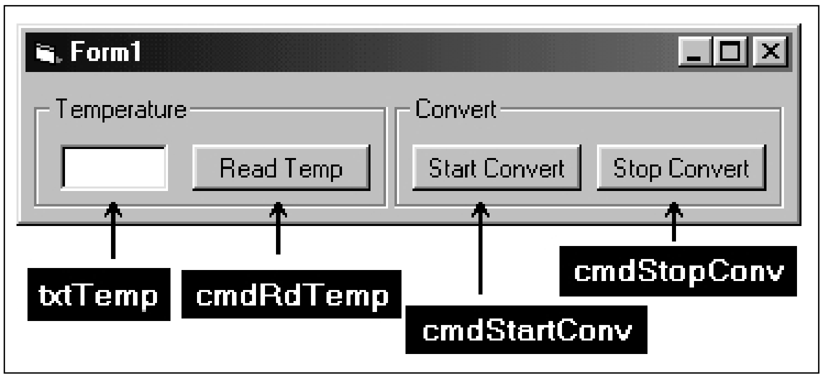


Figure P1-1 Temperature reader program form developed from Visual BASIC

2. Change properties of all control as :

Name	cmdRdTemp
Caption	Read Temp

Name	txtTemp
Text	

Name	cmdSTARTConv
Caption	Start Convert

Name	cmdSTOPConv
Caption	Stop Convert

3. Write VB code for **cmdSTARTConv_Click** enevt as :

```
Private Sub cmdSTARTConvert_Click()
    I2CSTART
    Send8BIT &H90
    Ack
    Send8BIT &HEE           'Start Convert Temperature Command
    Ack
    I2CSTOP
End Sub
```

4. Write VB code for **cmdSTOPConv_Click** event as :

```
Private Sub cmdSTOPConvert_Click()
    I2CSTART
    Send8BIT &H90
    Ack
    Send8BIT &H22           'Stop Convert Temperature Command
    Ack
    I2CSTOP
End Sub
```

5. Write VB code for **cmdRdTemp_Click** event as :

```
Private Sub cmdRdTemp_Click()
Dim tmp As Double
Dim datH As Integer
Dim datL As Integer
    I2CSTART
    Send8BIT &H90
    Ack
    Send8BIT &HAA           'Read Temperature Command
    Ack
    I2CSTART
    Send8BIT &H90 + 1
    Ack
    datH = Read8Bit         'Read TH Register
    MAck
    datL = Read8Bit         'Read TL Register
    MNAck
    I2CSTOP
    If (datL And &H80) = &H80 Then
        tmp = datH + 0.5
    Else
        tmp = datH
    End If
    txtTemp.Text = tmp
End Sub
```

6. After DS1621 interface process finish, must bring the result data to calculate addition. Verify bit 7 of low byte of Temperature register as ì1î or not. If yes, add 0.5 to high byte data of Temperature register. If not, the value of high byte data of Temperature register will be the target temperature value.

7. Connect P-Board with EX-10 board by I^{2}C bus cable.

8. Run program. Click Read Temp button

Reading data is 0. Because the conversion still not gegin.

9. Click Start Convert button to start conversion. Clock Read Temp button again. The temperature will appear on screen.

10. Give heat to DS1621 (by hand touching). Click Read Temp button again.

If nothing to change : *It means DS1621 does not convert temperature continuous. Because1SHORT bit in Configuration register as “1” means convert one time.*

If changing continuous *: It means 1Shot bit in Configuration register as “0”. DS1621 is set to convert continuous after receive Start Convert signal.*

Full sourcecode can see in LAB15A.VBP file in Parallel_port/Lab/Lab15 folder within INEXís Computer Interface CD-ROM.

Access DS1621ís Config register

Accessing DS1621ís Config register has 2 process. One is reading, another is writing. The read process has steps as :

(1) Send START condition.

(2) Send DS1621ís address on EX-10 board and clear LSB bit as ì0î to write mode.

(3) Wait for ACK condition from DS1621.

(4) Send command &HAC to access Config register.

(5) Wait for ACK condition from DS1621.

(6) Send START condition.

(7) Send DS1621ís address on EX-10 board and set LSB bit to read mode.

(8) Wait for ACK condition from DS1621.

(9) Get data from Config register.

(10) Send Master Not ACK condition to DS1621.

(11) Send STOP condition.

For accessing to write data to Config register has steps as :

(1) Send START condition.

(2) Send DS1621ís address and clear LSB bit to write mode.

(3) Wait for ACK condition from DS1621.

(4) Send command &HAC to access Config register.

(5) Wait for ACK condition from DS1621.

(6) Write data to Config register.

(7) Wait for ACK condition from DS1621.

(8) Send STOP condition.

10. Build 2 buttons and 1 TextBox addition following the figure P1-2.

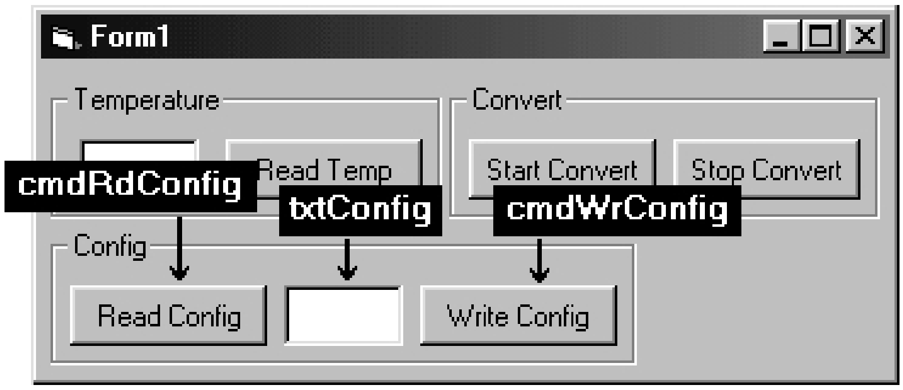


Figure P1-2 Temperature reader program that add command to read and write DS1621's Config register.

11. Change properties of some control as :

Name	cmdRdConfig
Caption	Read Config

Name	cmdWrConfig
Caption	Write Config

Name	txtConfig
Text	

12. Add program of **cmdRdConfig_Click** event to read data from Config regisrter

```
Private Sub cmdRdConfig_Click()
    I2CSTART
    Send8BIT &H90
    Ack
    Send8BIT &HAC
    Ack
    I2CSTART
    Send8BIT &H91
    Ack
    txtConfig.Text = Hex(Read8Bit)
    MNAck
    I2CSTOP
End Sub
```

13. Add program of **cmdWrConfig_Click** event for writing data to Config register

```
Private Sub cmdWrConfig_Click()
    I2CSTART
    Send8BIT &H90
    Ack
    Send8BIT &HAC
    Ack
    Send8BIT CByte("&H" & txtConfig.Text)
    Ack
    I2CSTOP
End Sub
```

14. Clear **1Shot** bit in DS1621ís Config register for setting to continuous mode by type number 8 into **txtConfig** box and click **Write Config** button. After that click **Stop Convert** button (stop DS1621 to change conversion mode). Click **Start Convert** button to read temperature again in continuous mode.

Full sourcecode can see in LAB15B.VBP file in Parallel_port/Lab/Lab15 folder within INEXís Computer Interface CD-ROM.

DS1621 in Thermostat mode experiment

In this mode has 2 registers to setting the thermal temp points. One is TH for high temperature limit and TL for low temperature limit.

Reading value of TH and TL register has same procedure with Temperature register but different command. Read TH use &HA1, for TL use &HA2

Writing procedure include :

(1) Send START condition.

(2) Send DS1621ís address and clear LSB bit to write mode.

(3) Wait for ACK condition from DS1621.

(4) Send command &HA1 for TH register or &HA2 for TL register.

(5) Wait for ACK condition from DS1621.

(6) Send MSB bit of TH or TL register to DS1621.

(7) Wait for ACK condition from DS1621.

(8) Send LSB bit of TH or TL register to DS1621.

(9) Wait for ACK condition from DS1621.

(10) Send STOP condition.

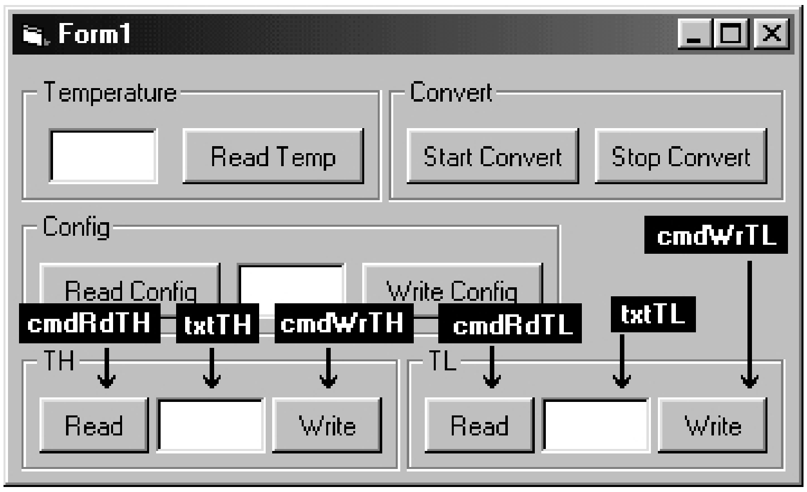


Figure P1-3 Shows the DS1621 Thermostat experimental program's screen.

15. Add control and change name following the figure P1-3

16. Add program of **cmdRdTH_Click** event as :

```
Private Sub cmdRdTH_Click()
Dim tmp As Double
Dim datH As Integer
Dim datL As Integer
    I2CSTART
    Send8BIT &H90
    Ack
    Send8BIT &HA1            'Access TH Register Command
    Ack
    I2CSTART
    Send8BIT &H91
    Ack
    datH = Read8Bit          'Read MSB TH Register
    MAck
    datL = Read8Bit          'Read LSB TH Register
    MNAck
    I2CSTOP
    If (datL And &H80) = &H80 Then
        tmp = datH + 0.5
    Else
        tmp = datH
    End If
    txtTH.Text = tmp
End Sub
```

*Note : Code of **cmdRdTL_Click** event is similar **cmdRdTH_Click** event. Different is only command code as &HA2 and store into txtTL.Text.*

17. Write code for **cmdWrTH_Click** event as :

```
Private Sub cmdWrTH_Click()
Dim tmp As Double
Dim datH As Byte
Dim datL As Byte
    datH = Fix(txtTH.Text)
    tmp = Val(txtTH.Text) - datH
    If tmp <> 0 Then datL = &H80
    I2CSTART
    Send8BIT &H90
    Ack
    Send8BIT &HA1           'Access TH Register Command
    Ack
    Send8BIT datH
    Ack
    Send8BIT datL
    Ack
    I2CSTOP
End Sub
```

cmdWrTL_Click *event is edited for receive data from txtTL.Text file and use command code &HA2*

18. Run program. Write Config to A, click Write Config button for setting conversion mode to continuous and Active High Thermostat mode.

19. Put number 30 into TH box and 28 into TL box. Click Write button of TH and TL

20. Click Start Convert button to start. Give heat to DS1621 by hand touching. Click Read Temp. See the reading temperature.

If value more than 30 *: LED on EX-10 board will light, beep generate and Relay active (if select jumper). Config register value will be 2AH because THF bit is set from temperature valuse reach over TH point.*

Full sourcecode can see in LAB15B.VBP file in Parallel_port/Lab/Lab15 folder within INEXís Computer Interface CD-ROM.