NetMedia

Basic Express
Application Note

Using a 74HCT251 8 Bit Input Multiplexer with BasicX

Introduction

This application note illustrates how to use a 74HCT251 input multiplexer with a BasicX system. The multiplexer allows you to effectively expand the number of input lines available on a processor. The chip also functions as an I/O buffer.

To control the chip, you need a minimum of 4 I/O lines from a BasicX system, consisting of 3 address lines and 1 data line. An additional output enable line can optionally be used to enable or disable the chip. The device provides 8 input lines.

Hardware interface

Figure 1 illustrates the pinouts for the device (Radio Shack part number 276-2864):

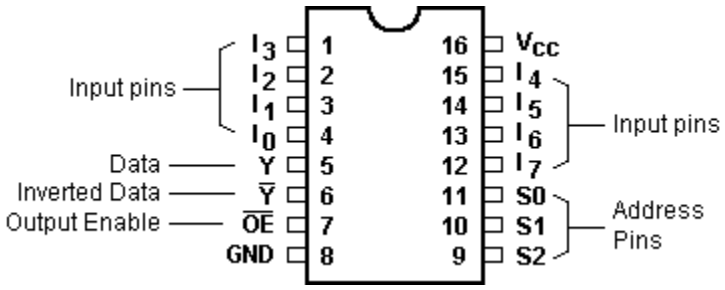

Figure 1 – 74HCT251 pinout

Figure 2 shows you how to connect the chip to a BX-24 system. Other BasicX systems use similar connections, except that pin numbers may be different:

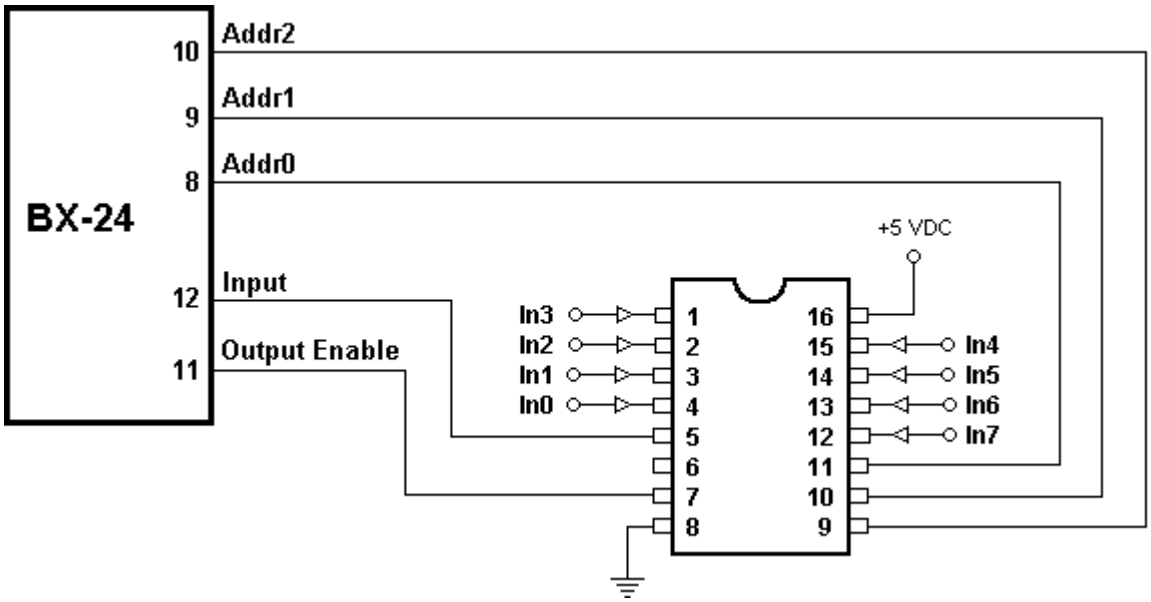

Figure 2 -- schematic

Operation

Reading an input is very simple – first, the output enable pin is held low to enable the chip, then a 3 bit address is written to address lines Addr0, Addr1 and Addr2. The last step is to read the input pin.

In some cases, it may be convenient to leave the chip permanently enabled, in which case you can tie the output enable line to ground, which reduces the number of I/O lines needed from the processor.

Software Interface

The attached demonstation program IMuxDemo reads and displays all 8 input pins. Most of the low level processing is done by the following procedures, which are located in module IMux74HCT251:

Procedure	Function
`InitializeIMux`	`Configures input pin to input-tristate, clears address pins, enables chip`
`TerminateIMux`	`Clears address pins, disables chip`
`GetPinIMux`	`Reads the input pin`
`PutAddressIMux`	`Writes 3 bit address to the address pins`

Program listing. The program IMuxDemo consists of the following 2 modules, which are included as separate files along with this application note:

IMuxDemo.bas -- Main program

IMux74HCT251.bas -- API for chip

Each file is listed on the following pages.

The program is written for a BX-24 system. If you want to port the program to other BasicX systems, most changes should be limited to renumbering the I/O pins in module IMux74HCT251.

Module IMuxDemo

```
Attribute VB_Name = "IMuxDemo"
'--------------------------------------------------------------------------------
Option Explicit

' This program reads a 74HCT251 input multiplexer chip and displays the
' state of each of the 8 input lines. The display is updated at about 2 Hz.
'--------------------------------------------------------------------------------
Public Sub Main()

    Dim State As Boolean
    Dim Address As Byte

    Debug.Print
    Debug.Print "Demo of 74HCT251 Input Multiplexer"
    Debug.Print

    Call InitializeIMux

    Do
        For Address = 0 To 7

            State = GetPinIMux(Address)

            If (State = True) Then
                Debug.Print "1 ";
            Else
                Debug.Print "0 ";
            End If
        Next

        ' Add <CR><LF>
        Debug.Print

        Delay 0.5
    Loop

End Sub
'--------------------------------------------------------------------------------
```

Module IMux74HCT251

```
Attribute VB_Name = "IMux74HCT251"
'--------------------------------------------------------------------------------
Option Explicit

' Driver for 74HCT251 input multiplexer (Radio Shack P/N 276-2864).

'////////// BX-01 Pin Assignments ///////////
'>>Private Const Addr0Pin        As Byte = 38
'>>Private Const Addr1Pin        As Byte = 37
'>>Private Const Addr2Pin        As Byte = 36
'>>Private Const OutputEnablePin As Byte = 35
'>>Private Const InputPin        As Byte = 34
'///////////////////////////////////////////

'////////// BX-24 Pin Assignments ///////////
Private Const Addr0Pin        As Byte =  8
Private Const Addr1Pin        As Byte =  9
Private Const Addr2Pin        As Byte = 10
Private Const OutputEnablePin As Byte = 11
Private Const InputPin        As Byte = 12
'///////////////////////////////////////////

'////////// BX-35 Pin Assignments ///////////
'>>Private Const Addr0Pin        As Byte = 25
'>>Private Const Addr1Pin        As Byte = 24
'>>Private Const Addr2Pin        As Byte = 23
'>>Private Const OutputEnablePin As Byte = 22
'>>Private Const InputPin        As Byte = 21
'///////////////////////////////////////////

Private Const EnableIMux      As Byte = 0
Private Const DisableIMux     As Byte = 1
'--------------------------------------------------------------------------------
Public Sub InitializeIMux()

    Call PutPin(InputPin, bxInputTristate)

    Call PutAddressIMux(0)

    Call PutPin(OutputEnablePin, EnableIMux)

End Sub
'--------------------------------------------------------------------------------
Public Sub TerminateIMux()

    Call PutAddressIMux(0)

    Call PutPin(OutputEnablePin, DisableIMux)

End Sub
'--------------------------------------------------------------------------------
```

```
'--------------------------------------------------------------------------------
Public Function GetPinIMux( _
    ByVal Address As Byte) As Boolean

    Call PutAddressIMux(Address)

    If (GetPin(InputPin) = 0) Then
        GetPinIMux = False
    Else
        GetPinIMux = True
    End If

End Function
'--------------------------------------------------------------------------------
Private Sub PutAddressIMux( _
    ByVal Address As Byte)

' Copy the lower 3 bits to the 3 address pins.

    Const Bit0 As Byte = bx00000001
    Const Bit1 As Byte = bx00000010
    Const Bit2 As Byte = bx00000100

    If ((Address And Bit0) = Bit0) Then
        Call PutPin(Addr0Pin, bxOutputHigh)
    Else
        Call PutPin(Addr0Pin, bxOutputLow)
    End If

    If ((Address And Bit1) = Bit1) Then
        Call PutPin(Addr1Pin, bxOutputHigh)
    Else
        Call PutPin(Addr1Pin, bxOutputLow)
    End If

    If ((Address And Bit2) = Bit2) Then
        Call PutPin(Addr2Pin, bxOutputHigh)
    Else
        Call PutPin(Addr2Pin, bxOutputLow)
    End If

End Sub
'--------------------------------------------------------------------------------
```


2.01.A