NetMedia

## Basic Express Application Note

# Interfacing Buttons and Switches

### Introduction

This application note describes a number of hardware and software methods for interfacing buttons and switches to a BasicX system.

The need to receive information or user input from the outside is an important part of most microcontroller applications. For example, a set of buttons could be used to power-up and set a BasicX-controlled milling machine. At the same time, limit switches could allow the BasicX system to sense that a protective cover has been removed, and the processor could thereby shut down the machine for safety reasons.

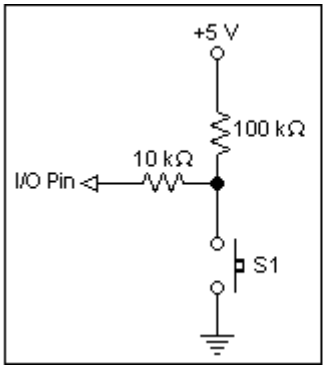


**Figure 1**

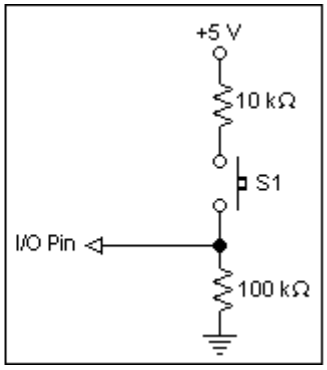


**Figure 2**

### Hardware interface

There are many ways to read the state of a button or switch. In this application note we will focus on the two most common methods. The first method (shown in figure 1) involves referencing the I/O pin high through a high value resistor. A switch pulls the I/O pin low through a second smaller resistor. This method will work equally well for both normally-open and normally-closed switches.

The second method (shown in figure 2) involves referencing the I/O pin to ground or low through a relatively high value resistor. A 100 kΩ resistor is used in the example diagram. The button is then used to pull the I/O pin high through a smaller resistor. Again, this method works equally well for both normally-open and normally-closed switches.

## Example Programs

This first example program assumes that you're using I/O pin 16 of a BasicX system and have a switch wired according to the diagram in Figure 1. The program reads the state of any switch connected to the pin and writes that value to a second pin (I/O pin 17, in this case). The logical state of the second I/O pin can then be verified with an oscilloscope or logic probe. An LED can also be used (see figure 3).

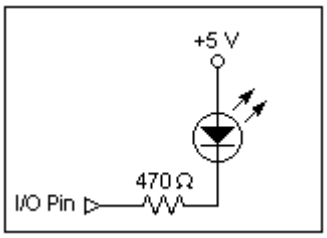


**Figure 3**

Example program 1:

```
Sub Main()

    Dim State as Byte

    Do
        ' Read I/O pin 16.
        State = GetPin(16)

        ' Copy the state to pin 17.
        Call PutPin(17, State)
    Loop

End Sub
```

This program works well with switches, but if you tried this example using a momentary button you probably noticed that pin 17 only stays in the momentary state as long as the button is pressed. This program would not work very well if you wanted to toggle the state of an I/O pin and have it stay there without having to hold down the button.

In order to make use of momentary buttons or switches a different approach is needed. This example program assumes that you are using pin 16 for the input and pin 17 for the output and have a momentary push button wired according to the diagram in Figure 1. Example program 2 will use the bitwise binary Xor operation to toggle the state of pin 17 each time the momentary button is pressed.

Example program 2:

```
Sub Main()

' This program reads the state of a switch and toggles the
' state of an output pin whenever the switch is pressed.

    Const InputPin As Byte = 16
    Const OutputPin As Byte = 17

    ' Configure pins.
    Call PutPin(InputPin, bxInputTristate)
    Call PutPin(OutputPin, bxOutputHigh)

    Dim State as Byte

    State = 0

    Do
        ' Toggle State if switch is pressed.
        If GetPin(InputPin) = 0 Then

            State = State Xor 1

            ' Pause a quarter-second for button de-bounce.
            Call Delay(0.25)
        End If

        ' Write State to output pin.
        Call PutPin(OutputPin, State)
    Loop

End Sub
```

A similar example program is contained in a separate file called Buttons.bas.



2.00.A