```
/*
A demonstration program for GCBASIC.
-------------------------------------------------------
-------------------------
Mechanical switches play an important and extensive
role in practically every computer, microprocessor and
microcontroller application.

Mechanical switches are inexpensive, simple and
reliable. However, switches can be very noisy
electrically.
The apparent noise is caused by the closing and opening
action that seldom results in a  clean electrical
transition.
The connection makes and breaks several, perhaps even
hundreds, of times before the final switch state
settles.
The problem is known as switch `bounce`. Some of the
intermittent activity is due to the switch contacts
actually bouncing off each other.
Imagine slapping two billiard balls together. The hard
non-resilient material doesn’t absorb the kinetic
energy of motion.
Instead, the energy dissipates over time and friction
in the bouncing action against the forces push the
billiard balls together.
Hard metal switch contacts react in much the same way.
Also, switch contacts are not perfectly smooth. As the
contacts move against each other, the imperfections and
impurities on the surfaces cause the electrical
connection to be interrupted.
The result is switch `bounce`.

The consequences of uncorrected switch bounce can range
from being just annoying to catastrophic.
For example, imagine advancing the TV channel, but
instead of getting the next channel, the selection
```

skips one or two.

This is a situation a designer should strive to avoid.

Switch bounce has been a problem even before the earliest computers. The classic solution involved filtering, such as through a resistor-capacitor circuit, or through re-set-
table shift registers (see Figure 3-4 and Figure 3-5, PDF 40001296b.pdf).
These methods are still effective but they involve additional cost in material, installation and board real estate.

Debouncing in software eliminates these additional costs. This is the purpose of this demonstration.

****

The demonstration use a function that examines the state of the button and returns one of four values. You can test the value and determine the correct action.

The values are BUTTON_UP, BUTTON_PRESSED, BUTTON_DOWN or BUTTON_RELEASED. The function also includes a debounce by using a wait to determine if the switch is still depressed.

```
@author    EvanV
@license   GPL
@version   1.00
@date      2024-08-17
******************************************************
```

```
*************************
*/


#chip 16F887
#option explicit

/*
           ------------PORTA----------------
    Bit#:  -7---6---5---4---3---2---1---0---
    IO:    ----------------------------AN0--
    IO:    ---------------------------------

           ------------PORTB----------------
    Bit#:  -7---6---5---4---3---2---1---0---
    IO:    ----------------------------SW---
    IO:    ---------------------------------

           ------------PORTC----------------
    Bit#:  -7---6---5---4---3---2---1---0---
    IO:    ---------------------------------
    IO:    ---------------------------------

           ------------PORTD----------------
    Bit#:  -7---6---5---4---3---2---1---0---
    IO:   -DS8-DS7-DS6-DS5-DS4-DS3-DS2-DS1--
    IO:    ---------------------------------

*/

DIR PORTD OUT
PORTD.7 = 1

DIR PORTB.0 In
#define SWITCH PORTB.0

Do
```

```
       // when the switch is down, then, process
       If switch_event = BUTTON_DOWN Then

              // Ensure the Carry bit is clear
              Set C OFF

              //Rotate the port to the right, shift the
bits of the port to the right
              ROTATE PORTD RIGHT

              //Did the rotate set the carry bit? If,
yes, set the bit to 1
              IF C = 1 Then PORTD.7 = 1

            'wait until the switch is release
            Wait Until switch_event = BUTTON_RELEASED

       End If

Loop

End



// Methods and subs

#DEFINE BUTTON_UP       0
#DEFINE BUTTON_PRESSED  1
#DEFINE BUTTON_DOWN     2
#DEFINE BUTTON_RELEASED 3
#DEFINE BUTTON_UNKNOWN  4



'/*******************************************************
```

```
***********************
'  Function:
'    input_event()
'
'  Summary:
'    Processes the single button into the states UP,
DOWN, PRESSED & RELEASED.
'
'  Description:
'    This function helps write user interface state
machines by determining when
'    the button was pressed, released
'
'  Precondition:
'    None
'
'  Parameters:
'    None
'
'  Returns:
'    value of the current button events.
'    Valid responses are BUTTON_UP, BUTTON_DOWN,
BUTTON_PRESSED, BUTTON_RELEASED
'
'  Remarks:
'       state_switch inverts the port. If high then use
state_switch=off

'       #define SWITCH PORTB.0
'       Dir SWITCH In
        #DEFINE STATE_SWITCH OFF
'
'
*******************************************************
********************/
```

```
function switch_event()

    Dim previous_switch_state  as Byte
    Dim current_switch_state as Byte

    current_switch_state = input_switch

   if !current_switch_state & !previous_switch_state
then
      ' button is not pressed now nor was it pressed
previously
        switch_event = BUTTON_UP
    END if
    if current_switch_state & !previous_switch_state
then
            ' button is pressed now but it wasn't
previously
        switch_event = BUTTON_PRESSED
    End if
    if current_switch_state & previous_switch_state
then
            ' button was pressed previously and is
still pressed
        switch_event = BUTTON_DOWN
    end if
    if !current_switch_state & previous_switch_state
 then
            ' button is not pressed now but it was
previously
        switch_event = BUTTON_RELEASED
    End If

    previous_switch_state  = current_switch_state

End Function
```

```
' Debounce button, Debounce switch
' This works by examination of port define by the
constant SWITCH
' If the SWITCH has been held down for 15 ms then the
SWITCH is pushed.
Function input_switch (  )

    Dim ButtonCount as byte

    input_switch  = false

    If SWITCH = STATE_SWITCH Then
        ButtonCount = 0
        Do While SWITCH = STATE_SWITCH and ButtonCount
< 4
            wait 5 ms
            ButtonCount += 1
        Loop
    end if
    If ButtonCount > 3 then
        input_switch  = true
        ButtonCount = 0
    end if

End Function
```