```
/*
A demonstration program for GCBASIC.
-------------------------------------------------------------------------
The demonstration combines previous demonstrations
using the button to reverse the direction of rotation.

When the button is pressed and adjusting the
potentiometer to control the speed of rotation.

The program needs to keep track of rotation direction
and new code needs to be added to rotate in the other
direction.

In the demonstration we will need to rotate left and
right, and, therefore check for a ‘1’ in bit 7 of the
LED port.

When the ‘1’ shows up in bit 7 of the display, insert
it into the bit 0 position.


@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

DIR PORTB.0 In
#define SWITCH PORTB.0

Dim ADCValue as Byte
Dim LEDsValue as Byte

LEDsValue = 128

#define LEFT_DIRECTION  1
#define RIGHT_DIRECTION 0

Dim direction as Bit
direction = RIGHT_DIRECTION
```

```
Do

    ADCValue = Scale(  ReadAD10 ( AN0 ), 0, 1023, 1,
250)

    //Wait for the value of ADC
    Wait ADCValue ms

    If switch_event = BUTTON_PRESSED Then

        If direction = RIGHT_DIRECTION Then
            direction = LEFT_DIRECTION
        Else
            direction = RIGHT_DIRECTION
        End If

    End if

    If direction = RIGHT_DIRECTION 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 LEDsValue RIGHT

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

    Else

        // Ensure the Carry bit is clear
        Set C OFF

        //Rotate the port to the left, shift the bits
```

```
of the port to the left
        ROTATE LEDsValue LEFT

        //Did the rotate set the top bit? If, yes, set
the carry
        IF C = 1 Then LEDsValue.0 = 1

    End If

    //Set the LEDs to the value of LEDsValue
    PORTD = LEDsValue

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
```