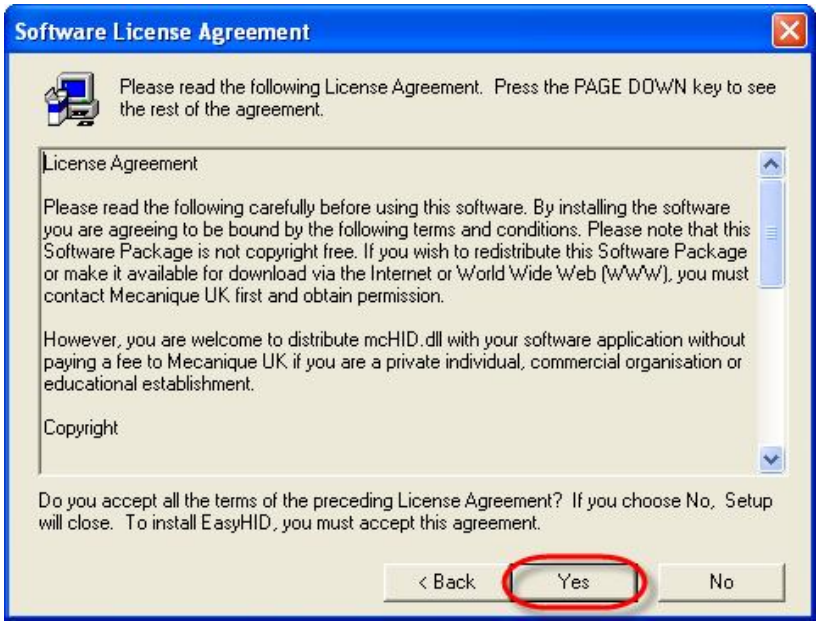
ตัวอย่างการพัฒนาโปรแกรม USB HID ของ PIC18F4550 ด้วย EasyHID

EasyHID เป็นชุดโปรแกรมเครื่องมือช่วยสนับสนุนการพัฒนา โปรแกรมไมโครคอนโทรลเลอร์ในตระกูล PIC18F เกี่ยวกับ USB HID โดยสนับสนุนการใช้งานร่วมกับไมโครคอนโทรลเลอร์ PIC18 ในรุ่นที่มีโมดูล USB (Universal Serial Bus) บรรจุไว้ภายในตัว MCU เช่น PIC18F2550, PIC18F4550 เป็นต้น โดยความสามารถของโปรแกรมตัวนี้คือ สามารถทำการสร้าง Source Code ที่เป็นฟังก์ชั่นการทำงานของ USB ในแบบ HID Device ให้เราได้อย่างง่ายดาย ทั้งฝั่งของไมโครคอนโทรลเลอร์และคอมพิวเตอร์ PC โดยในด้านของไมโครคอนโทรลเลอร์นั้นโปรแกรมจะสร้าง Source Code เป็นภาษาเบสิก (Basic Pro Compiler) ให้ ส่วนทางด้านฝั่งคอมพิวเตอร์ PC จะสามารถเลือกรูปแบบของ Source Code ได้ว่าจะเป็นของภาษา Visual Basic หรือ Visual C++ หรือ Delphi ทำให้เราสามารถลดระยะเวลาและขั้นตอนในการพัฒนาโปรแกรมสำหรับฟังก์ชั่น USB แบบ HID ลงไปได้เป็นอย่างมาก ช่วยลดขั้นตอนและความยุ่งยากซับซ้อนในการพัฒนาโปรแกรมได้เป็นอย่างมาก โดยวิธีการและขั้นตอนในการใช้ชุดโปรแกรม EasyHID เป็นเครื่องมือช่วยในการพัฒนาโปรแกรมสามารถศึกษาเรียนรู้ได้จากตัวอย่างต่อไปนี้

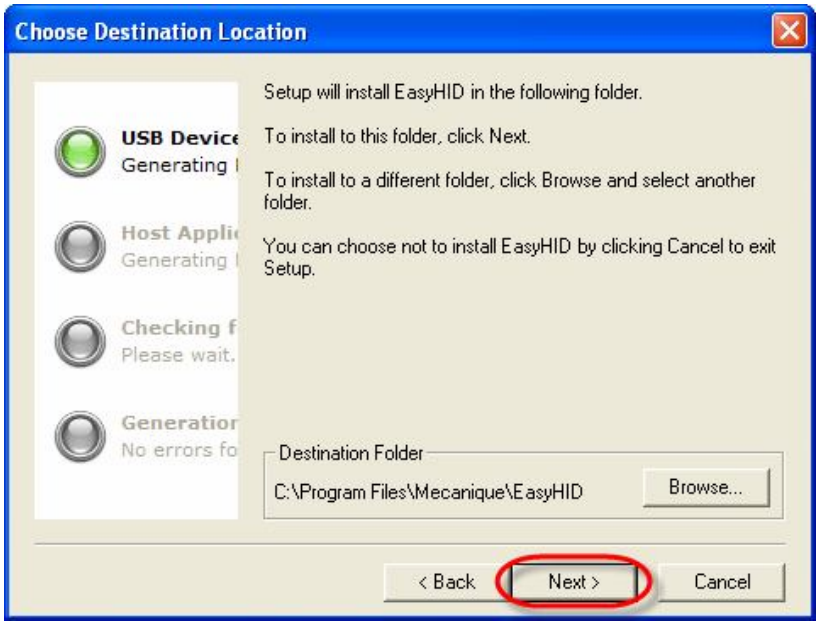
การติดตั้งโปรแกรม

1. เปิดโฟลเดอร์ EasyHID และ ดับเบิ้ลคลิก ไอคอนของไฟล์ SETUP เพื่อทำการเริ่มต้นกระบวนการติดตั้งโปรแกรมโดยโปรแกรมจะแสดงเมนูต้อนรับการติดตั้งให้เห็นจากนั้นให้เลือกNext ดังรูป

2. หลังจากเลือก Next โปรแกรมจะแสดงเงื่อนไขและข้อตกลงในการใช้งานโปรแกรมให้ทราบ ให้เลือก Yes เพื่อยอมรับเงื่อนไข ดังรูป

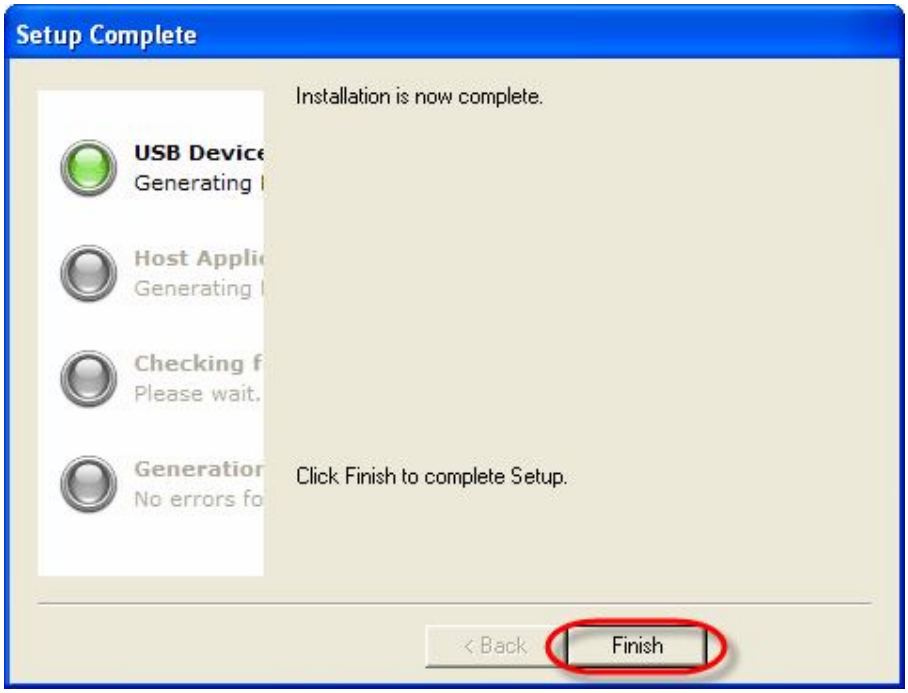


3. หลังจากยอมรับเงื่อนไขแล้วโปรแกรมจะเริ่มต้นทำการติดตั้งโปรแกรมทันที โดยในขั้นตอนนี้โปรแกรมจะให้เลือกตำแหน่งโฟลเดอร์ที่ต้องการติดตั้งโปรแกรม เพื่อความสะดวกให้เลือกตามค่า Default มาตรฐาน แล้วเลือก Next ดังรูป



4. มาถึงขั้นตอนนี้ โปรแกรมจะให้กำหนดตำแหน่ง Group ในการสร้างไอคอนเรียกใช้งานโปรแกรม ให้เลือกตามค่า Default มาตรฐานแล้วเลือก Next ดังรูป

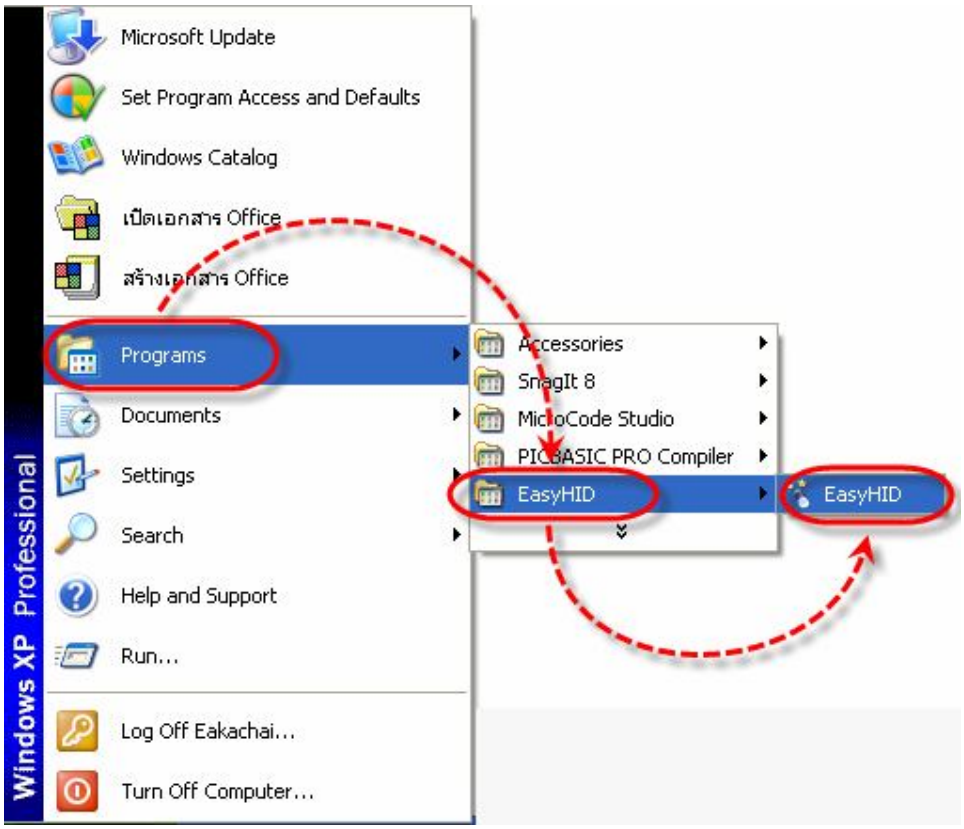
5. ในขั้นตอนนี้โปรแกรมจะเริ่มต้นทำการติดตั้งไฟล์ต่างๆลงในเครื่องคอมพิวเตอร์ ให้รอจนกระบวนการติดตั้งเสร็จสมบูรณ์ แล้วเลือก Finish เป็นอันเสร็จสิ้นขั้นตอนการติดตั้งโปรแกรม ดังรูป



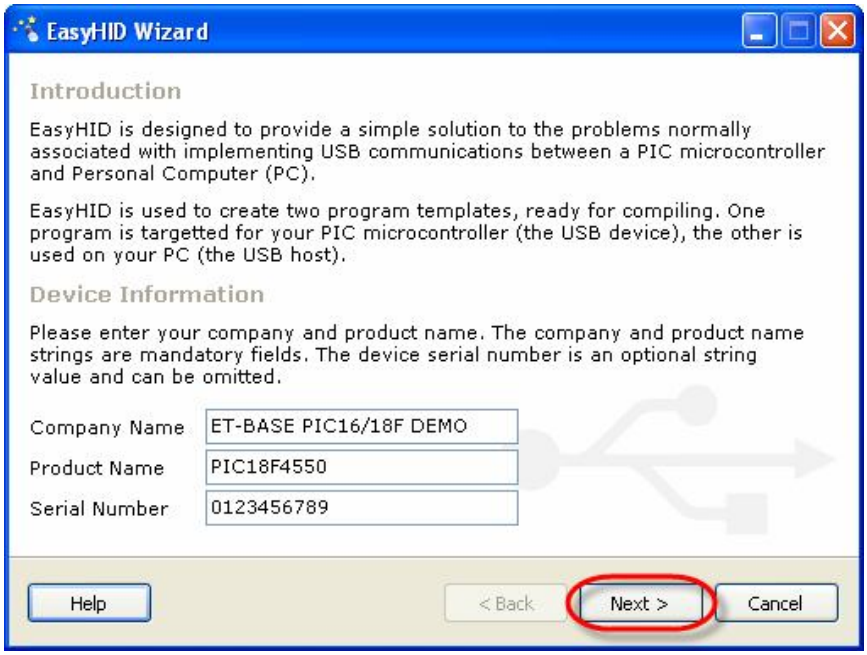
ตัวอย่างการใช้โปรแกรม EasyHID สร้าง Source Code

หลังจากทำการติดตั้งโปรแกรมเสร็จเรียบร้อยแล้ว ขั้นตอนต่อไปก็คือ การเรียกใช้งานโปรแกรมเพื่อกำหนดให้โปรแกรมสร้าง Source Code ให้ โดยในที่นี้จะแสดงลำดับขั้นตอนในการพัฒนาโปรแกรม HID ของไมโครคอนโทรลเลอร์เบอร์ PIC18F4550 โดยใช้ระบบฮาร์ดแวร์ของบอร์ด ET-BASE PIC16/18F ซึ่งทำการติดตั้งใช้งานกับ MCU เบอร์ PIC18F4550 โดยจะกำหนดฟังก์ชั่นการทำงานของ USB เป็นแบบ HID เพื่อทำการติดต่อสั่งงานไมโครคอนโทรลเลอร์ผ่านทางคอมพิวเตอร์ PC โดยทางด้านไมโครคอนโทรลเลอร์จะพัฒนาโปรแกรมด้วยภาษาเบสิก (Basic Pro Compiler) และในส่วนของคอมพิวเตอร์ PC จะใช้การพัฒนาโปรแกรมด้วยภาษา Visual Basic โดยการทำงานของโปรแกรมตัวอย่างที่จะสร้างขึ้นนี้จะออกแบบให้คอมพิวเตอร์ PC สามารถสั่ง ON/OFF การ ติด-ดับ ของหลอดแสดงผล LED บนบอร์ด และในขณะเดียวกันก็จะอ่านค่าของการกดสวิตช์ และ อ่านค่าสัญญาณ Analog จาก ADC ขึ้นมาแสดงผลบนหน้าจอโปรแกรมบนคอมพิวเตอร์ PC ด้วย จะโดยมีลำดับขั้นตอนดังนี้

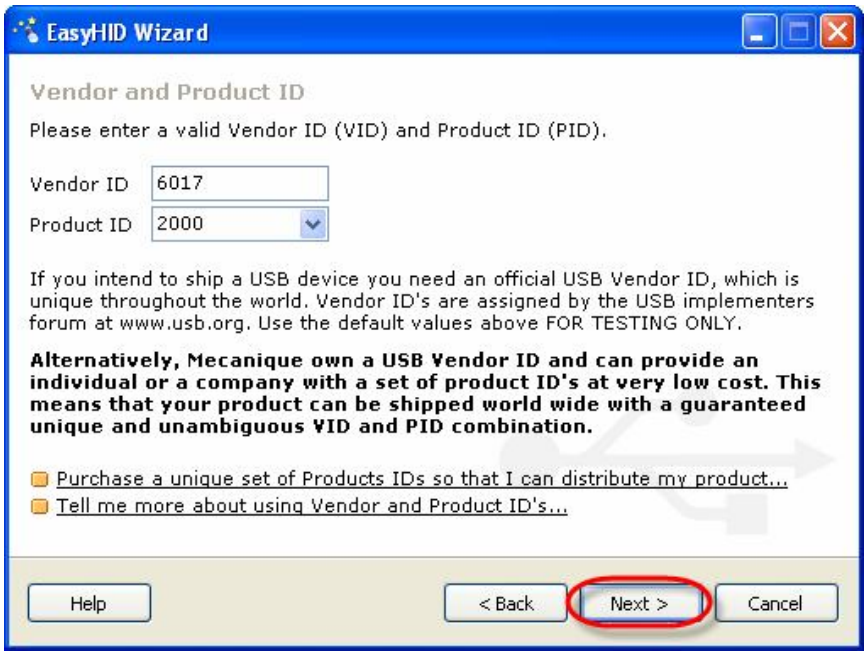
1. ทำการเรียกใช้โปรแกรม EasyHID โดยเลือก Start → Program → EasyHID → EasyHID



2. เมื่อโปรแกรมเริ่มทำงานจะปรากฏหน้าต่าง EasyHID Wizard ขึ้นมาให้เราทำการกำหนดค่าโดยการ ใส่ชื่อ Company Name, Product Name และ Serial Number ตามที่เราต้องการ ดังเช่นตัวอย่างในรูป จากนั้นเลือก Next ดังตัวอย่าง

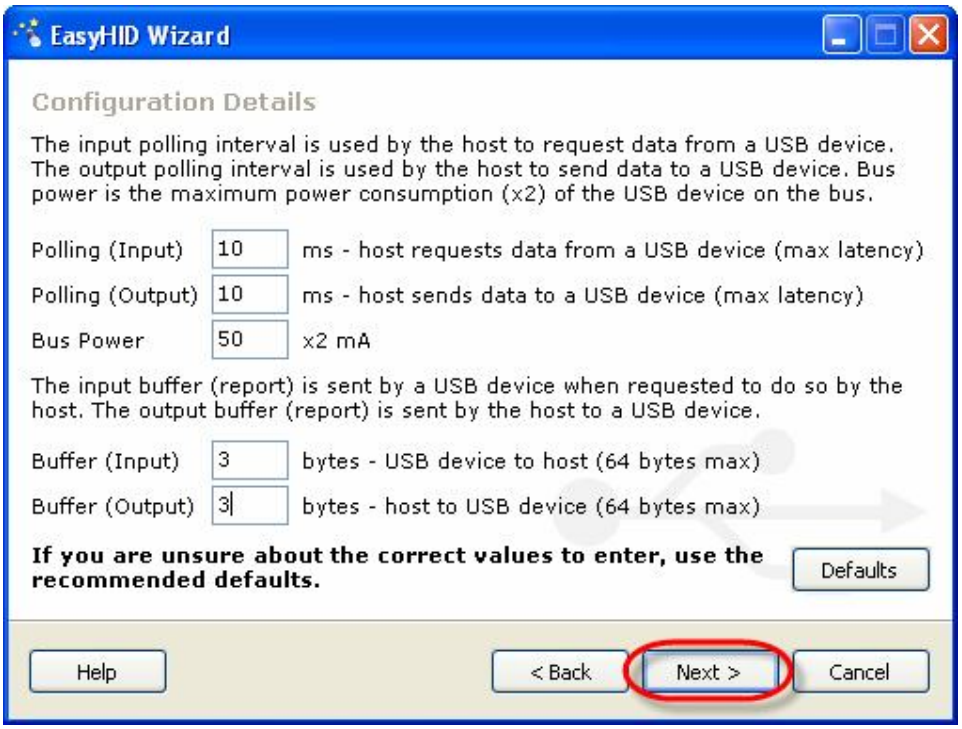


3. ในขั้นตอนนี้โปรแกรมจะรอให้กำหนดรหัส Vendor ID และ Product ID ของอุปกรณ์ USB ที่เราต้องการสร้างขึ้น ให้ทำการกำหนดค่าตามค่า Default ที่โปรแกรมกำหนดให้ แล้วเลือก Next ดังตัวอย่าง



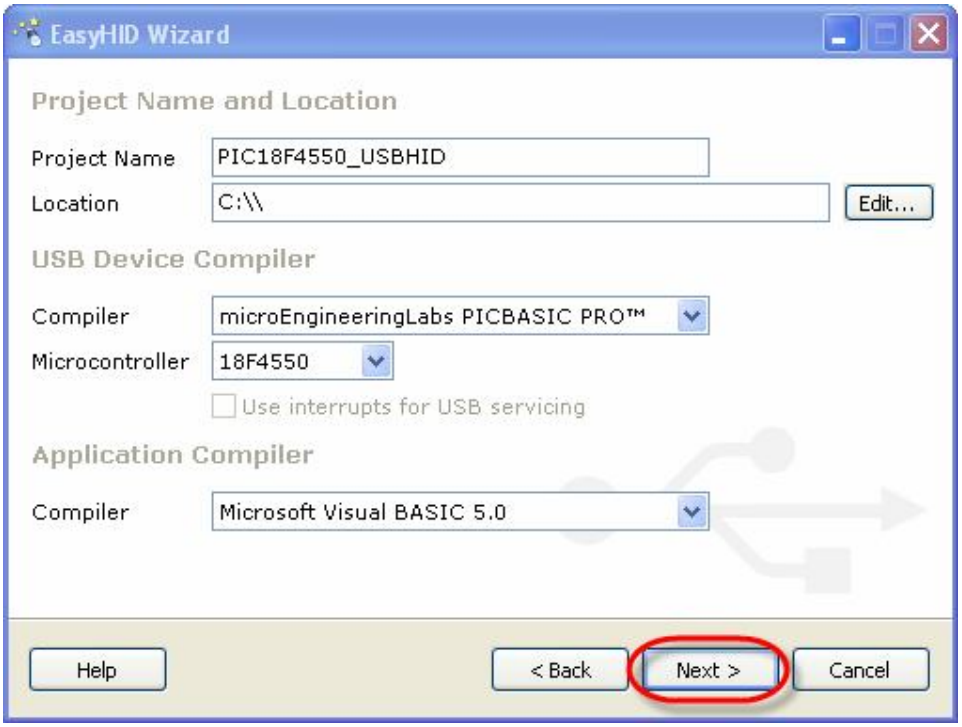
4. ในขั้นตอนนี้โปรแกรมจะให้กำหนดคุณสมบัติการสื่อสารของ USB HID หรือ Device ที่เราจะสร้างขึ้นมาโดยใช้ไมโครคอนโทรลเลอร์ PIC18F4550 ซึ่งค่าคุณสมบัติที่กำหนดนี้จะถูกนำไปเป็นสร้างเป็น Source Code ตามที่เรากำหนดไว้ด้วย ให้ทำการกำหนดค่าดังนี้คือ

- Polling Input เป็นค่าวงรอบเวลาในการร้องขอข้อมูลจาก USB Device ของ USB Host ให้กำหนดค่าเป็น 10mS
- Polling Output เป็นค่าวงรอบเวลาในการส่งข้อมูลจาก USB Host ไปให้ USB Device ให้กำหนดค่าเป็น 10mS
- Bus Power เป็นค่ากระแสที่กำหนดให้ USB Host จ่ายกระแสออกมาให้กับ USB Device ผ่านทางขั้วต่อ USB ซึ่งค่ากระแสสูงสุดที่ USB Host สามารถจ่ายให้ได้ คือ 500mA ในที่นี้ให้เรากำหนดเป็น 100mA โดยค่าที่กำหนดจะเป็นค่าที่นำไปคูณด้วย 2 ดังนั้นให้กำหนดค่าเป็น 50
- Buffer Input และ Buffer Output ให้กำหนดขนาดเป็น 3 Byte ซึ่งตามปรกติสามารถกำหนดได้สูงสุด 64 Byte แต่ในที่นี้เราจะกำหนดไว้เพียง 3 Byte ตามจุดประสงค์ที่เราต้องการใช้งาน โดยโครงสร้างการทำงานของ HID ที่ออกแบบไว้ในเบื้องต้น คือ
    - ไบต์ที่ 1 ใช้บรรจุค่าของ ADC (0..255)
    - ไบต์ที่ 2 ใช้บรรจุค่าของ Switch (0 = SW Press, 1 = SW Release)
    - ไบต์ที่ 3 ใช้บรรจุค่าของ LED(0 = LED OFF, 1 = LED ON)

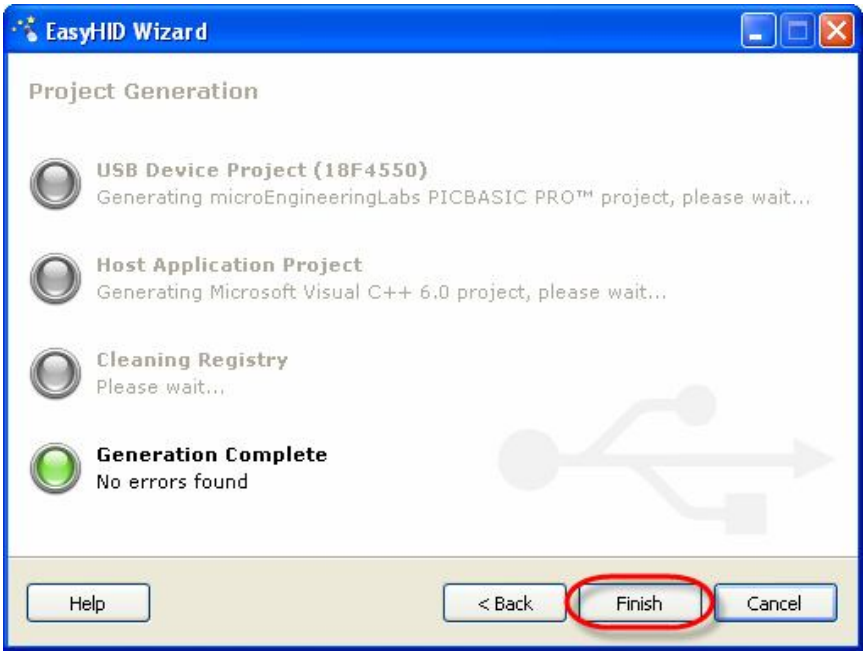


5. ในขั้นตอนนี้โปรแกรมจะให้กำหนดชื่อ Project Name และ ตำแหน่งที่ต้องการเก็บไฟล์ที่จะสร้างขึ้น พร้อมทั้งรูปแบบภาษาของ Source Code ที่ต้องการให้โปรแกรม EasyHID สร้างให้ ซึ่งโปรแกรม EasyHID จะสร้าง Source Code ได้ทั้งส่วนที่เป็นของไมโครคอนโทรลเลอร์ และ ส่วนของโปรแกรมสั่งงานสำหรับทำงานคอมพิวเตอร์PC ให้ด้วย โดยให้ทำการกำหนดค่าตัวเลือกต่างๆเป็นดังนี้

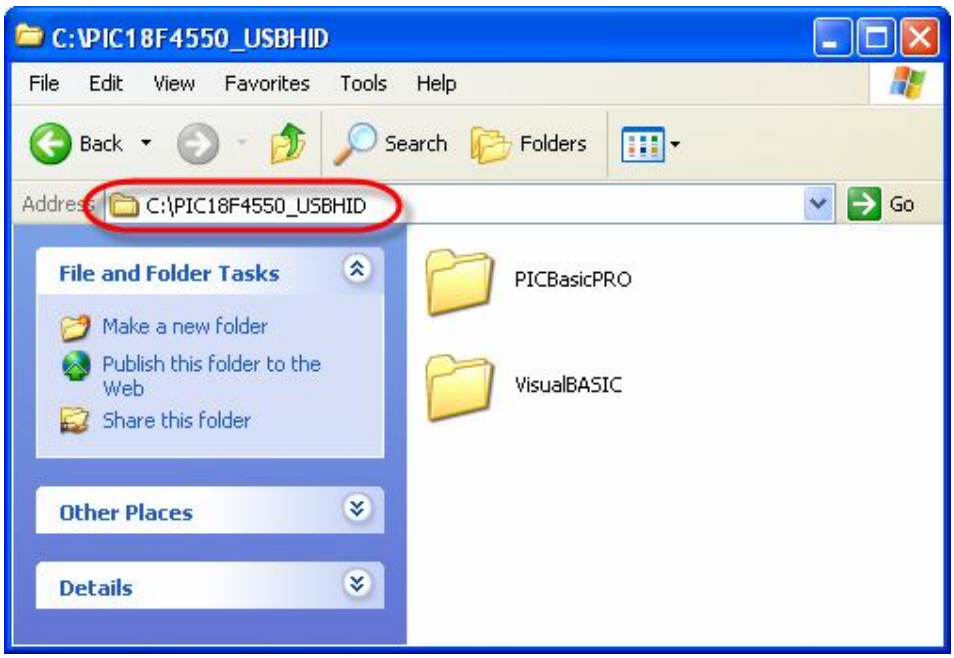
- Project Name and Location
  - Project Name กำหนดเป็น PIC18F4550_USBHID
  - Location กำหนดเป็น C:\\
- USB Device Compiler กำหนดเป็นภาษาเบสิก(PIC Basic Pro Compiler) สำหรับ PIC18F4550 ดังนั้นให้เลือกเป็น
  - Compiler กำหนดเป็น microEngineeringLabs PICBASIC PRO
  - Microcontroller กำหนดเป็น 18F4550
- Application Compiler
  - Compiler เลือกเป็น Microsoft Visual Basic 5.0



6. เมื่อทำการกำหนดคุณสมบัติต่างๆเสร็จเรียบร้อยแล้ว โปรแกรมจะทำการสร้าง Source Code ให้ ซึ่งจะใช้เวลาซักครู่ ให้รอจนโปรแกรมรายงานผล Generation Complete แล้วเลือก Finish ดังรูป

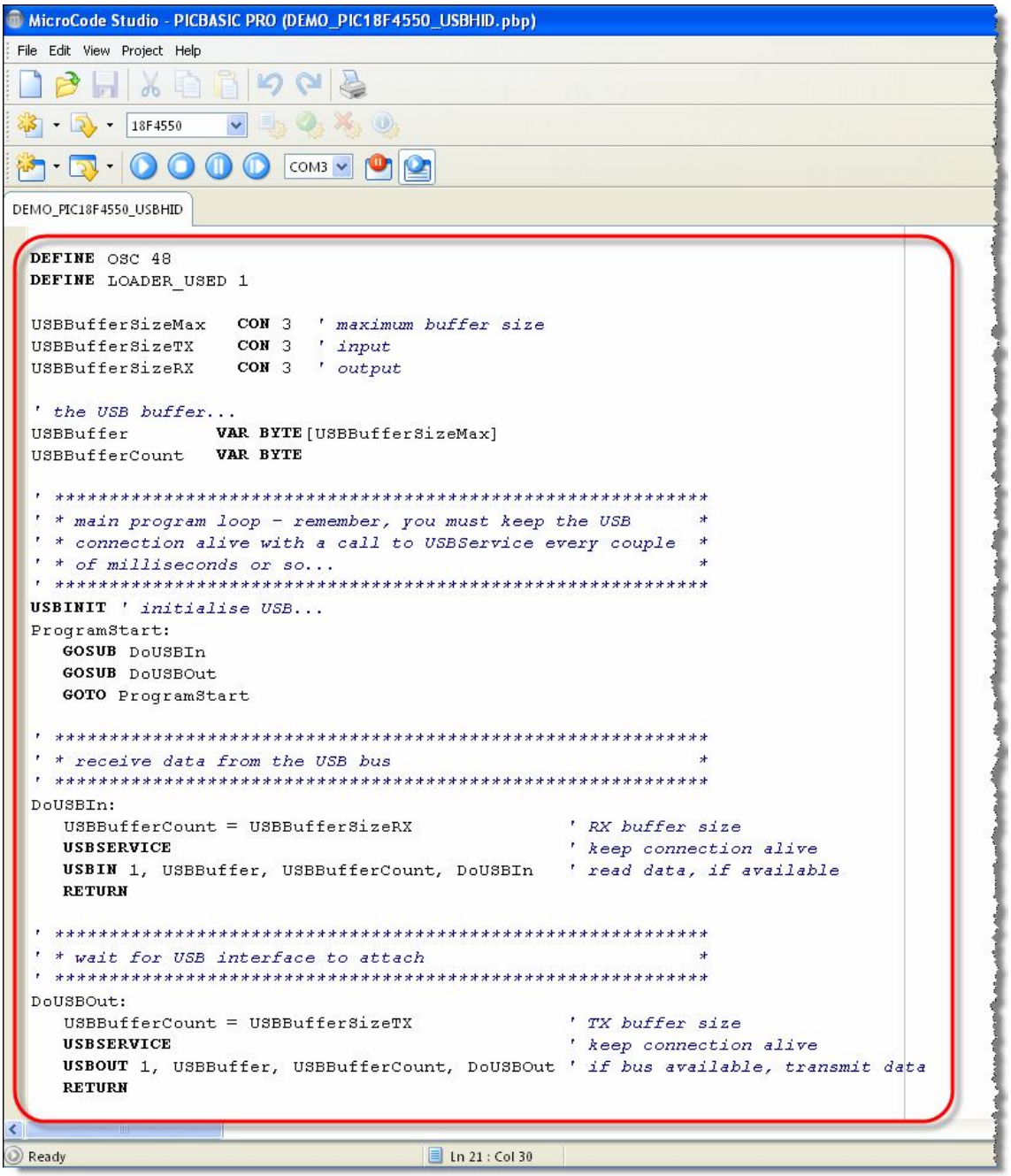


7. หลังจากผ่านขั้นตอนข้างต้น ถ้าไม่มีข้อผิดพลาดใดๆเกิดขึ้นโปรแกรม EasyHID จะทำการสร้าง Source Code ให้ตามเงื่อนไขที่เรากำหนดไว้ โดยจะมี Source Code ของโปรแกรมถูกสร้างขึ้นมา 2 ชุด โดยเป็นของ PICBasicPRO และ VisualBASIC ดังตัวอย่าง



โดยในส่วนของ Source Code ภาษาเบสิก(PIC Basic Pro Compiler) นั้น โปรแกรม EasyHID จะสร้าง Source Code พร้อมทั้งไฟล์ Library ที่เกี่ยวข้องขึ้นมาให้ โดยในส่วนของ File หลักนั้น ก็จะสร้างตัวแปร และ ฟังก์ชั่นย่อยในการเชื่อมต่อกับ USB HID จัดเตรียมไว้ให้เรียบร้อยแล้ว โดยมี 3 ฟังก์ชั่นหลักๆ คือ

- USBINIT เป็นส่วนของการ Initial การทำงานของ MCU และ USB Hardware ภายในตัว MCU
- DoUSBIn เป็นฟังก์ชั่นส่วนของการรอรับข้อมูล HID Packet จาก USB Host
- DoUSBOut เป็นฟังก์ชั่นส่วนของการจัดส่งข้อมูลผ่าน HID Packet ไปยัง Host



```
DEFINE OSC 48
DEFINE LOADER_USED 1

USBBufferSizeMax   con 3  'maximum buffer size
USBBufferSizeTX    con 3  'input
USBBufferSizeRX    con 3  'output

' the USB buffer...
USBBuffer          Var Byte[USBBufferSizeMax]
USBBufferCount     Var Byte

' ************************************************************
' * main program loop - remember, you must keep the USB      *
' * connection alive with a call to USBService every couple  *
' * of milliseconds or so...                                 *
' ************************************************************
usbinit ' initialise USB...

ProgramStart:

   gosub DoUSBIn
   gosub DoUSBOut

   goto ProgramStart

' ************************************************************
' * receive data from the USB bus                            *
' ************************************************************
DoUSBIn:
  USBBufferCount = USBBufferSizeRX              'RX buffer size
  USBService                                    'keep connection alive
  USBIn 1, USBBuffer, USBBufferCount, DoUSBIn   'read data, if available
  return

' ************************************************************
' * wait for USB interface to attach                         *
' ************************************************************
DoUSBOut:
  USBBufferCount = USBBufferSizeTX              'TX buffer size
  USBService                                    'keep connection alive
  USBOut 1, USBBuffer, USBBufferCount, DoUSBOut 'if bus available,send data
  return
```

แสดงตัวอย่าง Source Code ภาษาเบสิก ที่ได้จากโปรแกรม EasyHID

โดย Source Code ที่ได้จากโปรแกรม EasyHID นั้น ยังขาดความสมบูรณ์ เนื่องจากจะประกอบไปด้วยฟังก์ชั่นพื้นฐานในการสื่อสารรับส่งข้อมูลระหว่าง HID Device กับ USB Host เท่านั้น แต่ยังไม่มีส่วนของการตรวจสอบเงื่อนไขค่าของข้อมูลในPacket ต่างๆ ดังนั้น เราจะต้องทำการเพิ่มเติมเงื่อนไขการทำงานส่วนนี้เข้าไปเอง เพื่อให้การทำงานของโปรแกรมเป็นไปได้ตามจุดประสงค์ที่เราต้องการ

สำหรับรูปแบบการทำงานของ USBHID Device ที่ออกแบบไว้ในตัวอย่างนี้ จะกำหนดให้ไมโครคอนโทรลเลอร์ PIC18F4550 คอยตรวจสอบรอรับ Packet HID จาก USB Host แล้วนำข้อมูลใน Packet ที่ตรวจจับได้มาตรวจสอบเพื่อนำไปแปลเป็นเงื่อนไขในการใช้ควบคุมการ ON/OFF LED จากนั้นก็จะทำการอ่านค่า ADC และตรวจสอบ สถานะของสวิตช์ กดติด-ปล่อยดับ ว่าถูกกดหรือปล่อย แล้วแปลค่าของ ADC และ Switch เป็น HID Packet ส่งกลับออกไปยัง USB Host

โดยในส่วนของ Hardware I/O ของ PIC18F4550 นั้นกำหนดให้ใช้สัญญาณต่างๆดังนี้

- RA0 ทำหน้าที่เป็น ADC คอยอ่านค่าแรงดันที่ได้จากการปรับค่า Volume ปรับค่าได้ โดยกำหนดย่านการวัดของ ADC เป็นขนาด 8 บิตมีค่าระหว่าง 0 ถึง 255
- RA1 ทำหน้าที่เป็น Input Logic คอยอ่านค่าสถานะของสวิตช์ SW1 ว่าถูกกดหรือปล่อย โดยจะแปลงค่าเป็นข้อมูลระหว่าง 0(กด) และ 1(ปล่อย)
- RA2 ทำหน้าที่เป็น Output Logic โดยใช้ LED เป็นตัวแสดงค่าผลลัพธ์ทางโลจิก โดยให้ค่าข้อมูล 0 ในการสั่ง OFF LED และใช้ค่า 1 ในการสั่ง ON LED

ซึ่งจากเงื่อนไขการทำงานข้างต้น นั้นเราจะกำหนด Packet HID ของ USB ซึ่ง USB Buffer จะแบ่งออกเป็น 2 ชุด สำหรับใช้รับข้อมูล(DoUSBIn) และ ส่งข้อมูล(DoUSBOut) โดย USB Buffer จะมีขนาดเท่าๆกัน คือ 3 ไบต์ โดยเราจะกำหนดให้โครงสร้างข้อมูลใน Packet HID แต่ละไบต์เป็นดังนี้

- ไบต์ที่ 1 ใช้บรรจุค่าของ ADC ซึ่งมีค่าระหว่าง 0 ถึง 255
- ไบต์ที่ 2 ใช้บรรจุค่าสถานะสวิตช์กดติดปล่อยดับ โดยมีค่าระหว่าง 0 หรือ 1 โดยให้ 0 แทนความหมายสวิตช์ถูกกด และ ให้ 1 แทนความหมายของ สวิตช์ถูกปล่อย
- ไบต์ที่ 3 ใช้บรรจุรหัสควบคุมการ ติด ดับ ของ LED โดยมีค่าระหว่าง 0 หรือ 1 โดยให้ 0 แทนการดับ และ 1 แทนการติดของหลอดแสดงผล LED

โดยในด้านของการส่ง Packet HID เราจะต้องกำหนดให้โปรแกรมอ่านค่าของ ADC และ SW1 แล้วแปลงค่าเป็นข้อมูลบรรจุไว้ใน USB Buffer ไบต์ที่1 และ ไบต์ที่2 ตามลำดับ ส่วนค่าของ LED ON/OFF เราจะไม่สนใจใน Packet ของการส่งข้อมูล

ส่วนในด้านของการรับ Packet HID เข้ามานั้น เราก็จะไม่สนใจค่าของข้อมูลไบต์ที่1 และไบท์ที่2 แต่จะนำเอาเฉพาะค่าของข้อมูลไบต์ที่3 มาตรวจสอบว่าเป็น 1 หรือ 0 แล้วสั่ง ON หรือ OFF การแสดงผลของ LED ตามข้อมูลที่รับเข้ามาได้จาก Packet HID

ซึ่งเราจะต้องเพิ่มคำสั่งของโปรแกรม สำหรับใช้กำหนดการทำงานของขาสัญญาณ RA0(ADC), RA1(Input Logic ค่าของ SW1) และ RA2(Output Logic ขับ LED) และส่วนของการตรวจสอบค่าข้อมูลใน Packet HID เพิ่มเติมเข้าไปดังตัวอย่าง

```
DEFINE OSC 48
DEFINE LOADER_USED 1

USBBufferSizeMax   con 3                        'maximum buffer size
USBBufferSizeTX    con 3                        'input
USBBufferSizeRX    con 3                        'output

' the USB buffer...
USBBuffer          Var Byte[USBBufferSizeMax]
USBBufferCount     Var Byte

'*************************************************************
'* main program loop - remember, you must keep the USB      *
'* connection alive with a call to USBService every couple  *
'* of milliseconds or so...                                 *
'*************************************************************

usbinit ' initialise USB...

'Start of Initial ET-BASE PIC16/18F Hardware I/O
VR1 VAR PORTA.0                             'VR1 = RA[0]
TRISA.0 = 1                                 'ADC=Input
DEFINE ADC_BITS 8                           'Set number of ADC bits
DEFINE ADC_CLOCK 3                          'Set clock source (rc = 3)
DEFINE ADC_SAMPLEUS 50                      'Set sampling time(uS)
ADCON1 = %00001110                          'Only PORTA.0 is ADC input

SW1 VAR PORTA.1                             'SW1 = RA[1]
TRISA.1 = 1                                 'SW1=Input

LED var PORTA.2                             'LED = RA[2]
TRISA.2 = 0                                 'LED=Output
HIGH LED                                    'Default LED = ON
'End of Initial ET-BASE PIC16/18F Hardware I/O

ProgramStart:

   ADCIN 0, USBBuffer[0]                    'Read channel 0 to USBBuffer[0]
   while  ADCON0.1                          'Wait conversation completed
   wend


   if SW1 == 0 Then                         'Put SW1 Result to USBBuffer[1]
     USBBuffer[1] = 0
   else
     USBBuffer[1] = 1
   endif
   gosub DoUSBOut                           'Send USB Packet Data

   gosub DoUSBIn                            'Read Update USBBuffer
   if usbbuffer[2] = 0 then                 'Update USBBuffer[2] to LED
     low led
   else
     high led
   endif

   goto ProgramStart
```

```
'*************************************************************
'* receive data from the USB bus                             *
'*************************************************************
DoUSBIn:
  USBBufferCount = USBBufferSizeRX              'RX buffer size
  USBService                                    'keep connection alive
  USBIn 1, USBBuffer, USBBufferCount, DoUSBIn   'read data, if available
  return

'*************************************************************
'* wait for USB interface to attach                          *
'*************************************************************
DoUSBOut:
  USBBufferCount = USBBufferSizeTX              'TX buffer size
  USBService                                    'keep connection alive
  USBOut 1, USBBuffer, USBBufferCount, DoUSBOut 'if bus available,send data
  return
```

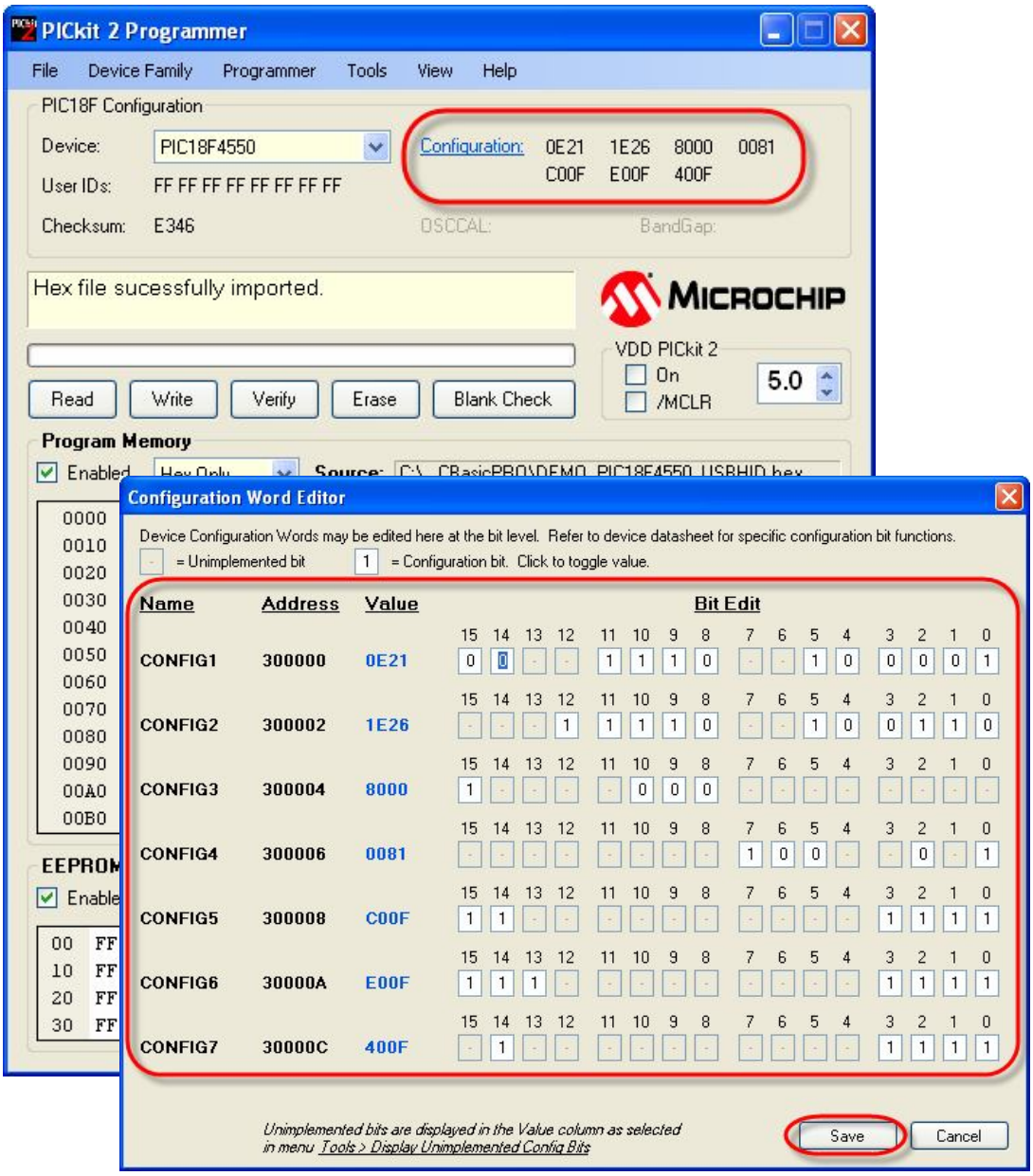
แสดง Code ภาษาเบสิก (Basic Pro) ที่ทำการแก้ไขเพิ่มเติมเรียบร้อยแล้ว

สำหรับในกรณีใช้ภาษาเบสิก(PIC Basic Pro Compiler) นั้น นอกจากส่วนของ Code ภาษาเบสิกที่เขียนขึ้นแล้ว ยังต้องมีการกำหนดค่า Configuration เพื่อเลือกกำหนดการทำงานให้กับ MCU ด้วยโดยกำหนดผ่านทางไฟล์ "C:\PBP\18F4550.INC"

```
'ET-BASE PIC16/18F Hardware used X-TAL=8MHz
'Setup Clock Config -> 8MHz/2=4MHz -> 4MHzxPLL(96MHz)->96MHz/2=48MHz
'Config1 = 0x0E21  = 0000 1110 : 0010 0001
'Config2 = 0x1E26  = 0001 1110 : 0010 0110
'Config3 = 0x8000  = 1000 0000 : 0000 0000
'Config4 = 0x0081  = 0000 0000 : 1000 0001
'Confgi5 = 0xC00F  = 1100 0000 : 0000 1111
'Config6 = 0xE00F  = 1110 0000 : 0000 1111
'Config7 = 0x400F  = 0100 0000 : 0000 1111

'Start of Config ET-BASE PIC16/18F Hardware For PIC18F4550 USBHID Demo
NOLIST
ifdef PM_USED
LIST
"Error: PM does not support this device.  Use MPASM."
NOLIST
else
LIST
LIST p = 18F4550, r = dec, w = -311, w = -230, f = inhx32
INCLUDE "P18F4550.INC" ; MPASM  Header
__CONFIG _CONFIG1L,_PLLDIV_2_1L&_CPUDIV_OSC1_PLL2_1L&_USBDIV_2_1L
__CONFIG _CONFIG1H,_FOSC_HSPLL_HS_1H & _FCMEN_OFF_1H&_IESO_OFF_1H
__CONFIG _CONFIG2L,_PWRT_ON_2L&_BOR_ON_2L&_BORV_0_2L&_VREGEN_ON_2L
__CONFIG _CONFIG2H,_WDT_OFF_2H & _WDTPS_32768_2H
__CONFIG _CONFIG3H,_CCP2MX_OFF_3H&_PBADEN_OFF_3H&_LPT1OSC_OFF_3H&_MCLRE_ON_3H
__CONFIG _CONFIG4L,_STVREN_ON_4L&_LVP_OFF_4L&_ICPRT_OFF_4L&_XINST_OFF_4L&_DEBUG_OFF_4L
__CONFIG _CONFIG5L,_CP0_OFF_5L&_CP1_OFF_5L&_CP2_OFF_5L&_CP3_OFF_5L
__CONFIG _CONFIG5H,_CPB_OFF_5H&_CPD_OFF_5H
__CONFIG _CONFIG6L,_WRT0_OFF_6L&_WRT1_OFF_6L&_WRT2_OFF_6L&_WRT3_OFF_6L
__CONFIG _CONFIG6H,_WRTB_OFF_6H&_WRTC_OFF_6H&_WRTD_OFF_6H
__CONFIG _CONFIG7L,_EBTR0_OFF_7L&_EBTR1_OFF_7L&_EBTR2_OFF_7L&_EBTR3_OFF_7L
__CONFIG _CONFIG7H,_EBTRB_OFF_7H
NOLIST
endif
LIST
```

แสดง การกำหนดค่า Configuration Bit สำหรับ PIC18F4550 ของบอร์ด ET-BASE PIC16/18F

ในกรณีใช้ Pickit2 เป็นเครื่องโปรแกรม ในขั้นตอนของการโปรแกรมถ้าสามารถตรวจสอบค่า Configuration ได้จากโปรแกรม ซึ่งถ้ากำหนดค่า Configuration ในไฟล์ “C:\PBP\18F4550.INC” ไว้ถูกต้องเมื่อโหลด Hex เข้ามาในโปรแกรม ค่า Configuration ที่แสดงบนหน้าต่างโปรแกรมควรได้ค่าตามที่กำหนดไว้ ซึ่งถ้าไม่ตรงหรือลืมกำหนดค่า Configuration ไว้ใน Source Code สามารถเข้าไปแก้ไขค่า Configuration ได้เองที่หน้าต่างโปรแกรมของ Pickit2 ดังตัวอย่าง

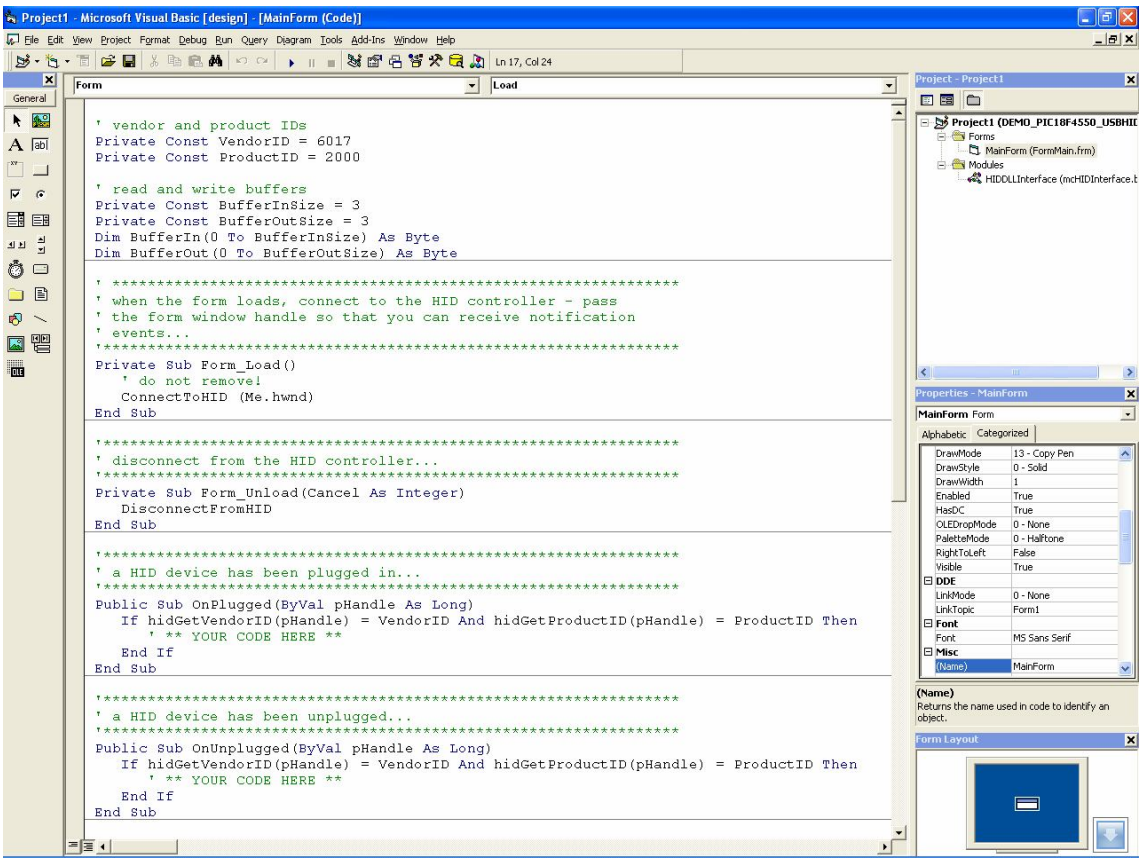


รูปแสดง ลักษณะของค่า Configuration ของ PIC18F4550 ของโปรแกรม Pickit2

การพัฒนา Source Code ของ Visual Basic

สำหรับ Source Code ของ Visual Basic ก็จะมีลักษณะเช่นเดียวกันคือ Source Code ที่ถูกสร้างขึ้นจากโปรแกรม EasyHID จะมีเฉพาะส่วนของโครงสร้างหลักๆของโปรแกรม สำหรับใช้ติดต่อสื่อสารกับไมโครคอนโทรลเลอร์ผ่านทาง USB HID ซึ่งประกอบด้วย การประกาศตัวแปรที่จำเป็นต้องใช้ และ สร้างฟังก์ชั่นในการ Initial และ รับ ส่ง ข้อมูล จัดเตรียมไว้ให้ ซึ่งเราจะต้องทำการเขียนโปรแกรมเพื่อกำหนดเงื่อนไขในการทำงานเพิ่มเติมเข้าไปตามจุดประสงค์ที่ต้องการเอง ไม่สามารถนำ Source Code ที่ได้มาใช้งานได้ทันที

เราต้องออกแบบสร้างฟอร์ม จัดวาง Component สำหรับติดต่อสื่อสารกับผู้ใช้ และเขียนโปรแกรมเชื่อมโยงเงื่อนไขการทำงาน การแสดงผลต่างๆ เพิ่มเติมเข้าไปเอง Source Code ที่ได้เป็นเพียงเครื่องมือที่จะใช้ในการสื่อสาร รับ ส่ง Packet ข้อมูลของ USB HID ระหว่าง USB Host ของคอมพิวเตอร์ PC กับไมโครคอนโทรลเลอร์เท่านั้น

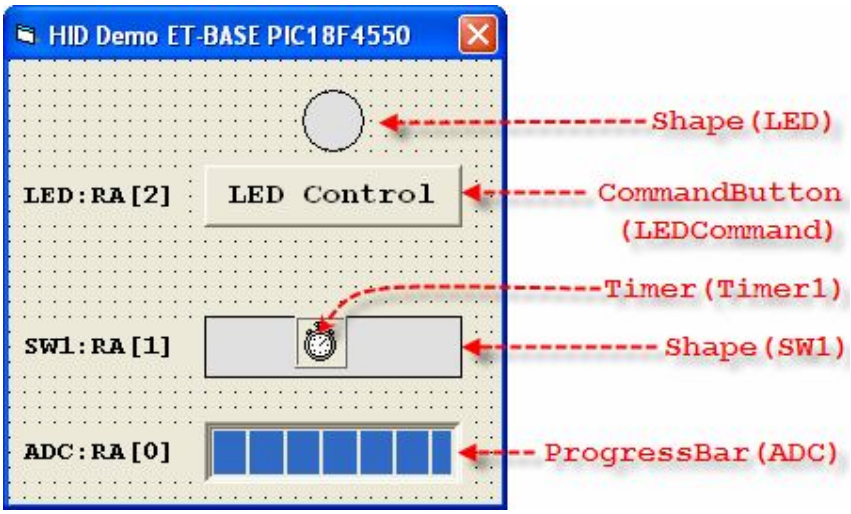


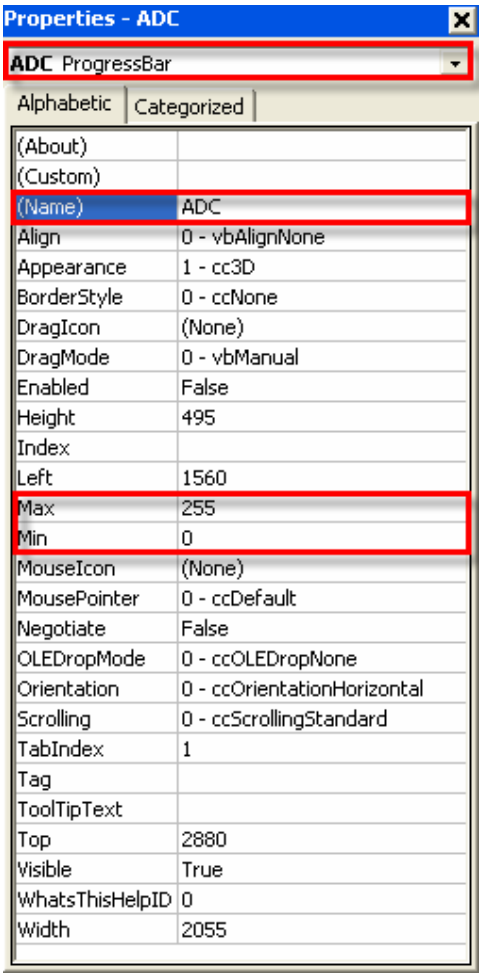
```
' vendor and product IDs
Private Const VendorID = 6017
Private Const ProductID = 2000

' read and write buffers
Private Const BufferInSize = 3
Private Const BufferOutSize = 3
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte

' ****************************************************************
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'****************************************************************
Private Sub Form_Load()
   ' do not remove!
   ConnectToHID (Me.hwnd)
End Sub

'****************************************************************
' disconnect from the HID controller...
'****************************************************************
Private Sub Form_Unload(Cancel As Integer)
   DisconnectFromHID
End Sub

'****************************************************************
' a HID device has been plugged in...
'****************************************************************
Public Sub OnPlugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle)
                              = ProductID Then
      ' ** YOUR CODE HERE **
   End If
End Sub

'****************************************************************
' a HID device has been unplugged...
'****************************************************************
Public Sub OnUnplugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle)
                              = ProductID Then
      ' ** YOUR CODE HERE **
   End If
End Sub

'****************************************************************
' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'****************************************************************
Public Sub OnChanged()
   Dim DeviceHandle As Long

   ' get the handle of the device we are interested in, then set
   ' its read notify flag to true - this ensures you get a read
   ' notification message when there is some data to read...
   DeviceHandle = hidGetHandle(VendorID, ProductID)
   hidSetReadNotify DeviceHandle, True
End Sub
```

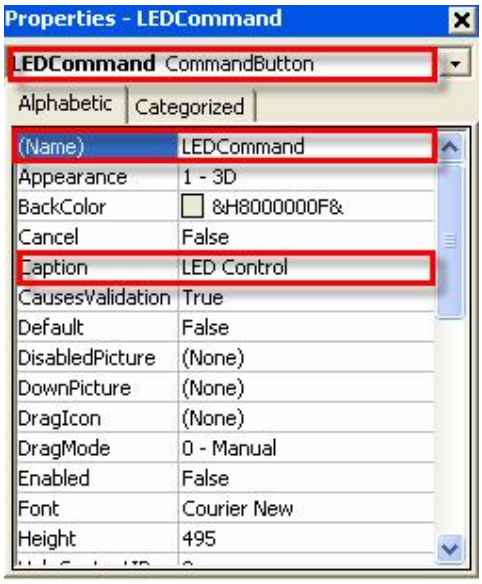
```
'*****************************************************************
' on read event...
'*****************************************************************
Public Sub OnRead(ByVal pHandle As Long)

   ' read the data (don't forget, pass the whole array)...
   If hidRead(pHandle, BufferIn(0)) Then
      ' ** YOUR CODE HERE **
      ' first byte is the report ID, e.g. BufferIn(0)
      ' the other bytes are the data from the microcontrolller...
   End If
End Sub

'*****************************************************************
' this is how you write some data...
'*****************************************************************
Public Sub WriteSomeData()
   BufferOut(0) = 0   ' first by is always the report ID
   BufferOut(1) = 10  ' first data item, etc etc

   ' write the data (don't forget, pass the whole array)...
   hidWriteEx VendorID, ProductID, BufferOut(0)
End Sub
```

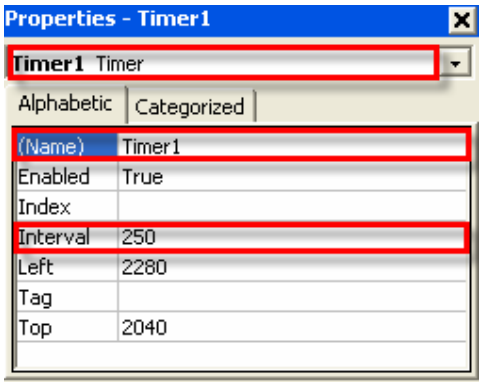
แสดง **Source Code** ของ **Visual Basic** ที่ได้จากโปรแกรม **EasyHID**

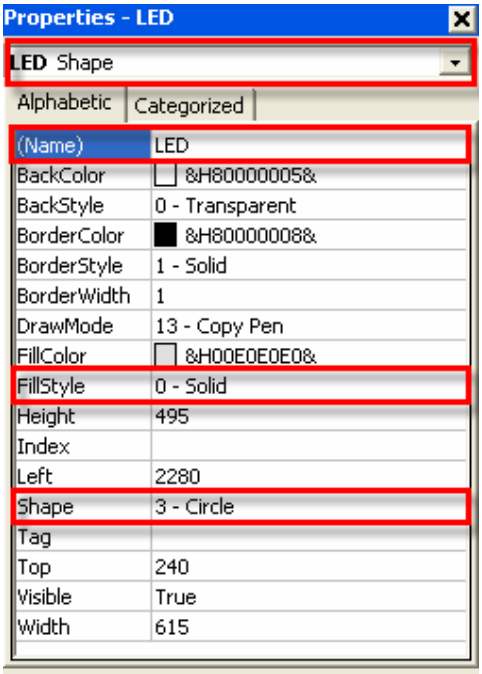
ซึ่งจาก Source Code ให้ทำการเปิดฟอร์ม ซึ่งจะเป็นฟอร์มเปล่าๆ ยังไม่มีการจัดวาง Component ใดๆไว้ให้ ให้ทำการจัดวาง Component และปรับแต่งดังตัวอย่าง

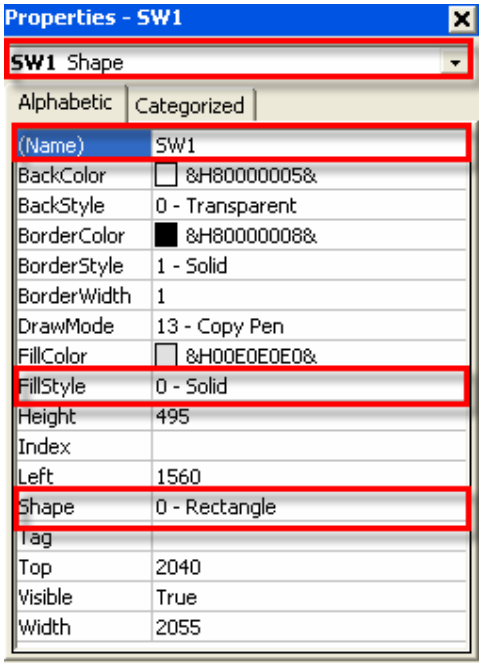












รูปแสดง การกำหนดปรับแต่งค่า Component บน Form ของ Visual Basic

หลังจากทำการจัดวาง Component บน Form และทำการปรับแต่งค่าของ Component เรียบร้อยแล้ว ก็ทำการเขียน Code เพื่อเชื่อมโยงเงื่อนไขการทำงานต่างๆเข้าด้วยกัน ซึ่งรูปแบบที่กำหนดไว้ จะใช้ CommandButton ชื่อ LEDCommand เป็นปุ่ม Button สำหรับใช้ส่งคำสั่ง ON/OFF การแสดงผล LED บนบอร์ด โดยให้ทำงานในลักษณะ Toggle คือ เมื่อมีการคลิกเมาส์ที่ปุ่ม จะสั่งให้ LED กลับสถานะเป็นตรงกันข้ามกับสถานะเดิมที่เป็นอยู่ เช่นถ้า OFF อยู่ก็จะเปลี่ยนเป็น ON และ ถ้า ON อยู่ก็จะเปลี่ยนเป็น OFF โดยค่าข้อมูลสำหรับสั่ง ON/OFF การแสดงผลของ LED จะถูกบรรจุไว้ใน Buffer ลำดับที่3 ของ Packet HID

และจะใช้ Timer ทำหน้าที่เป็นจับเวลาส่งค่าสถานะ Refresh ไปยังไมโครคอนโทรลเลอร์ เพื่อให้ไมโครคอนโทรลเลอร์ Echo ค่าสถานะของ สวิตช์ SW1 และค่าของ ADC มาให้ โดยเมื่อไมโครคอนโทรลเลอร์ Echo ข้อมูล Packet HID กลับเข้ามา โปรแกรมจะตรวจจับและจะตอบสนองการทำงานด้วยฟังก์ชั่น **OnRead** ซึ่งอยู่ใน **Public Sub OnRead(ByVal pHandle As Long)** โดยเราจะไปเขียนโปรแกรมเพื่อกำหนดเงื่อนไขการทำงานไว้ในฟังก์ชั่นนี้ โดยจะทำการอ่านค่าข้อมูลใน Packet และนำค่าใน ลำดับที่1 มาแปลงเป็นค่าแสดงผล ADC ให้กับ ProgressBar และนำค่าลำดับที่2 มาตรวจสอบพร้อมกับนำไปแสดงผล สถานะของสวิตช์ SW1 ด้วย Shape(SW1) โดยแสดงผลการทำงานเป็นค่าสี ให้ทราบว่า SW ถูกกดหรือปล่อย

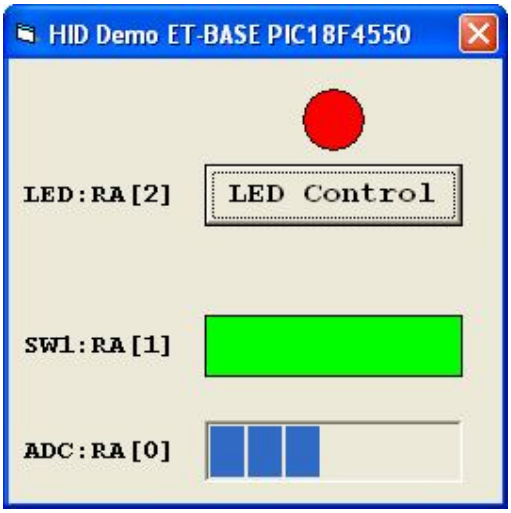
```
Const InputON = &HFF00&                                    ' Green Color
Const InputOFF = &H80000004                                ' Menu Color
Const OutputON = &HFF&                                     ' Red Color
Const OutputOFF = &HFFFFF8                                 ' Menu Color

' vendor and product IDs
Private Const VendorID = 6017
Private Const ProductID = 2000

' read and write buffers
Private Const BufferInSize = 3
Private Const BufferOutSize = 3
Dim BufferIn(0 To BufferInSize) As Byte
Dim BufferOut(0 To BufferOutSize) As Byte

' ****************************************************************
' when the form loads, connect to the HID controller - pass
' the form window handle so that you can receive notification
' events...
'****************************************************************
Private Sub Form_Load()

  ADC.Value = 0
  SW1.FillColor = InputOFF
  LED.FillColor = OutputOFF
  ADC.Enabled = False
  LEDCommand.Enabled = False

  ' do not remove!
  ConnectToHID (Me.hwnd)

End Sub

'****************************************************************
' disconnect from the HID controller...
'****************************************************************
Private Sub Form_Unload(Cancel As Integer)

   DisconnectFromHID

End Sub

'****************************************************************
' a HID device has been plugged in...
'****************************************************************
Public Sub OnPlugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle)
                              = ProductID Then

     ' ** YOUR CODE HERE **
     ADC.Enabled = True
     LEDCommand.Enabled = True

   End If
End Sub
```

```
'*****************************************************************
' a HID device has been unplugged...
'*****************************************************************
Public Sub OnUnplugged(ByVal pHandle As Long)
   If hidGetVendorID(pHandle) = VendorID And hidGetProductID(pHandle)
                              = ProductID Then

     ' ** YOUR CODE HERE **
     SW1.FillColor = InputOFF
     LED.FillColor = OutputOFF

     ADC.Enabled = False
     LEDCommand.Enabled = False

   End If
End Sub

'*****************************************************************
' controller changed notification - called
' after ALL HID devices are plugged or unplugged
'*****************************************************************
Public Sub OnChanged()

   Dim DeviceHandle As Long

   ' get the handle of the device we are interested in, then set
   ' its read notify flag to true - this ensures you get a read
   ' notification message when there is some data to read...
   DeviceHandle = hidGetHandle(VendorID, ProductID)
   hidSetReadNotify DeviceHandle, True

End Sub

'*****************************************************************
' on read event...
'*****************************************************************
Public Sub OnRead(ByVal pHandle As Long)

   ' read the data (don't forget, pass the whole array)...
   If hidRead(pHandle, BufferIn(0)) Then

     ' first byte is the report ID, e.g. BufferIn(0)
     ' the other bytes are the data from the microcontrolller...
     ' ** YOUR CODE HERE **

     '1st Byte Data = ADC(0..255) Value
     ADC.Value = BufferIn(1)

     '2nd Byte Data = SW1(0,1) Value
     If BufferIn(2) = 0 Then
       SW1.FillColor = InputON
     Else
       SW1.FillColor = InputOFF
     End If

   End If

End Sub
```

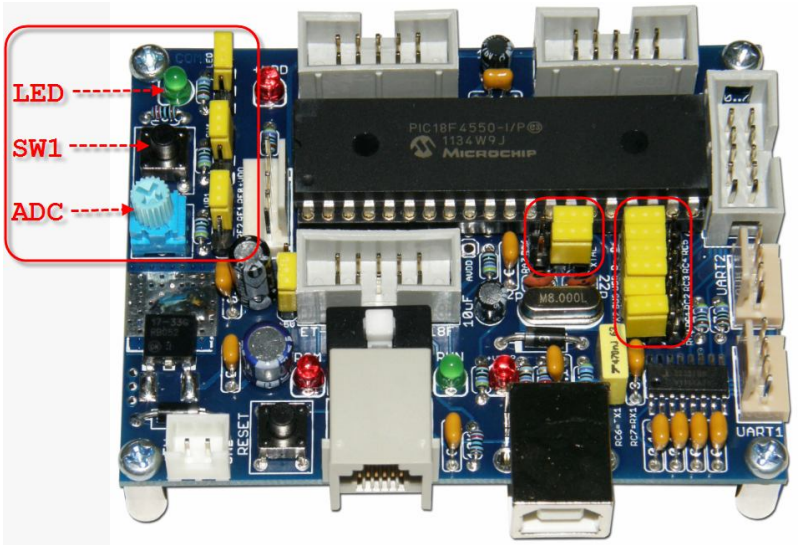
```
'*****************************************************************
' this is how you write some data...
'*****************************************************************
Public Sub WriteSomeData()

   BufferOut(0) = 0   ' first by is always the report ID
   BufferOut(1) = 10  ' first data item, etc etc

   ' write the data (don't forget, pass the whole array)...
   hidWriteEx VendorID, ProductID, BufferOut(0)

End Sub

'***************************
'* Send Toggle ON/OFF LED *
'***************************
Private Sub LEDCommand_Click()

  BufferOut(0) = 0                        'first by is always the report ID

  If LED.FillColor = OutputOFF Then
     LED.FillColor = OutputON
     BufferOut(3) = 1                     'Data ON LED
  Else
     LED.FillColor = OutputOFF
     BufferOut(3) = 0                     'Data OFF LED
  End If

  ' write the data (don't forget, pass the whole array)...
  hidWriteEx VendorID, ProductID, BufferOut(0)

End Sub

'250mS Trigger Update USB Data
Private Sub Timer1_Timer()

  BufferOut(0) = 0                        'first by is always the report ID

  'Refresh LED Status to Controller
  If LED.FillColor = OutputOFF Then
     BufferOut(3) = 0                     'Data OFF LED
  Else
     BufferOut(3) = 1                     'Data ON LED
  End If

  ' write the data (don't forget, pass the whole array)...
  hidWriteEx VendorID, ProductID, BufferOut(0)

End Sub
```

แสดง **Source Code** ของ **Visual Basic** ที่ทำการปรับปรุงแก้ไขเรียบร้อยแล้ว

เมื่อทำการ Compile และ Run จะได้ผลดังตัวอย่าง



↑↓



ในการทดสอบการทำงานของโปรแกรมต้องทำการเชื่อมต่อสาย USB เพื่อทำการเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ ET-BASE PIC16/18F(PIC18F4550) ที่ทำการโปรแกรม Hex File ที่ได้จาก Basic PRO Compiler เรียบร้อยแล้ว จากนั้นก็จะสามารถควบคุมสั่งงาน LED บนบอร์ดจากหน้าต่างโปรแกรมนี้ได้ทันที และเมื่อทดลอง กดสวิตช์ SW1 บนบอร์ด ก็จะเห็นสถานะของ SW1 บนหน้าจอโปรแกรมเปลี่ยนแปลงไปตามการกดและปล่อยสวิตช์ด้วย และในทำนองเดียวกัน ถ้าทำการปรับค่าตัวต้านทานปรับค่าดู ก็จะเห็นค่า การแสดงผลของ ProgressBar แสดงผลของ ADC เปลี่ยนแปลงตามไปด้วย