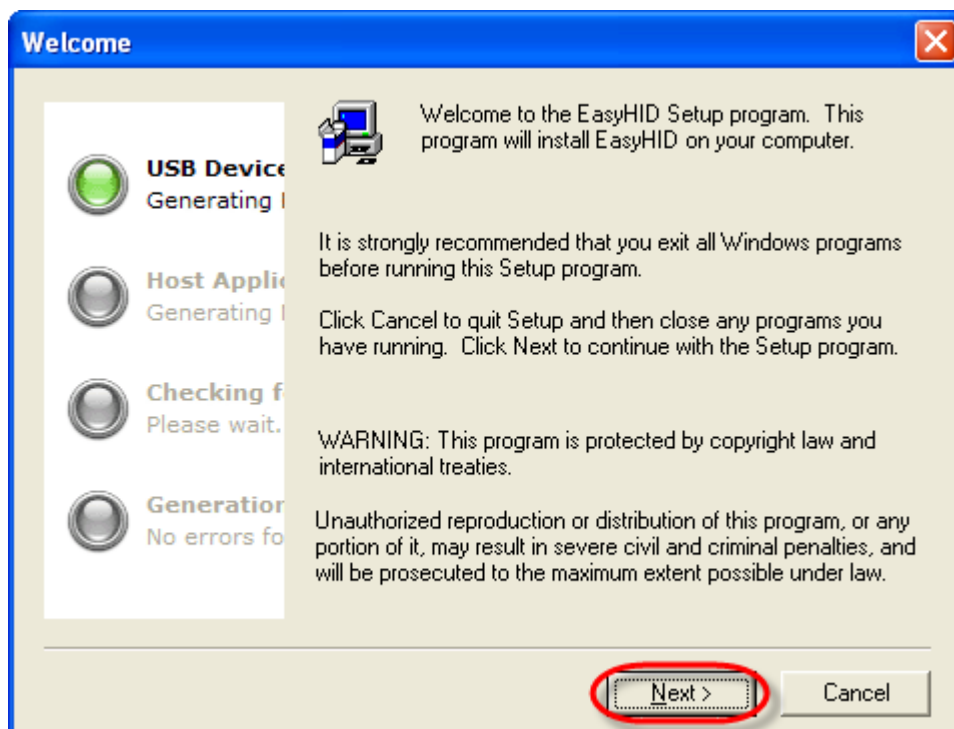


Example of Developing Program USB HID of PIC18F4550 by EasyHID

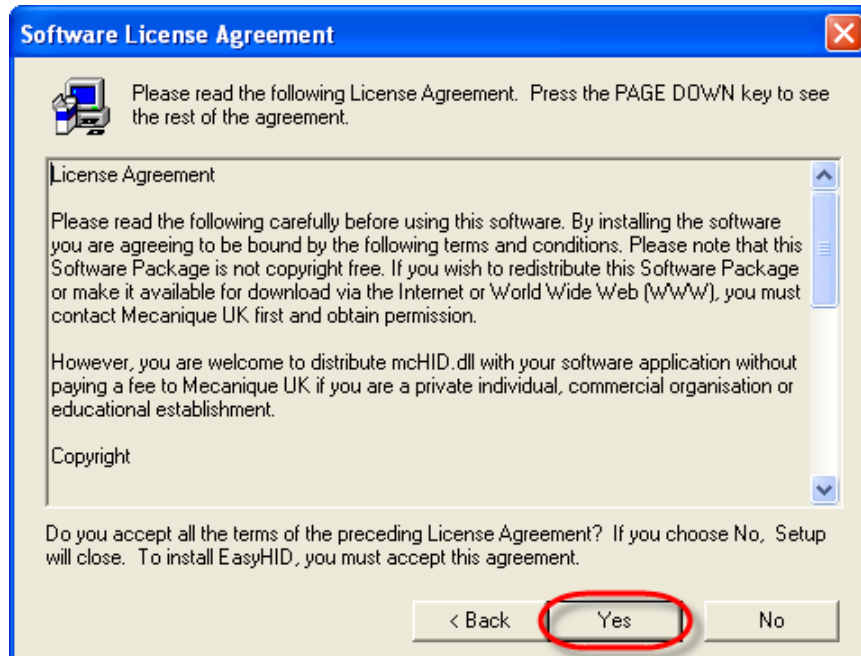
EasyHID is Programmer device that supports program development of Microcontroller in the series of PIC18F concerning USB HID; it supports the operation with PIC18 Microcontroller version that has included Module USB (Universal Serial Bus) internal MCU such as PIC18F2550, PIC18F4550, and etc. This program has capability to easily create Source Code that is the function of USB HID Device on both sides of Microcontroller and computer PC. On the side of Microcontroller, the program creates Source Code by BASIC Language (Basic Pro Compiler); on the side of computer PC, user can choose the preferable format of Source Code to be Visual Basic Language or Visual C++ Language or Delphi Language. So, it saves much time and reduces many procedures to develop program for Function USB HID; moreover, it reduces many procedures and much difficulty in developing program. User can refer following examples to learn the method and procedure for using this EasyHID Programmer to be tool for developing program;

How to install Program

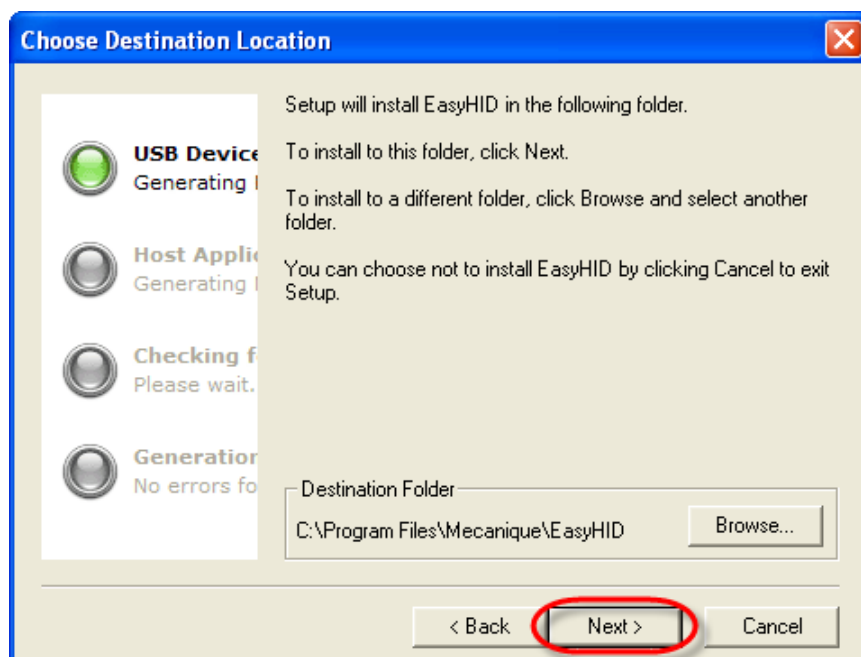
1. Open Folder EasyHID and double-click Icon **File SETUP**; it starts installing the program by showing the window Welcome. Then, click **Next** as shown in the picture below.



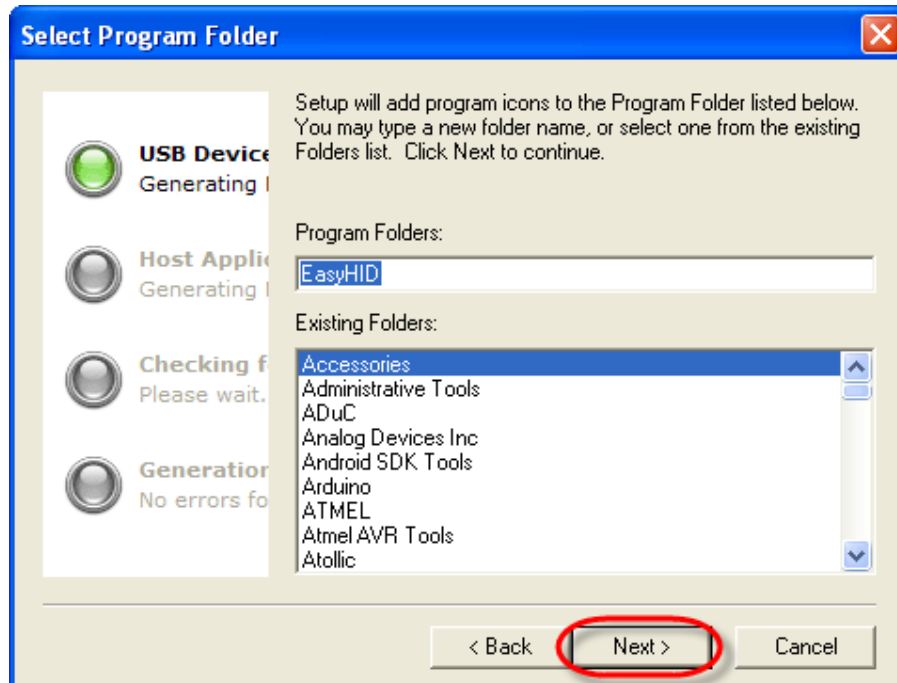
2. After clicked **Next**, program shows window Software License Agreement that is condition and agreement for using program; in this case, click **Yes** to accept the conditions and agreement as shown in the picture below;



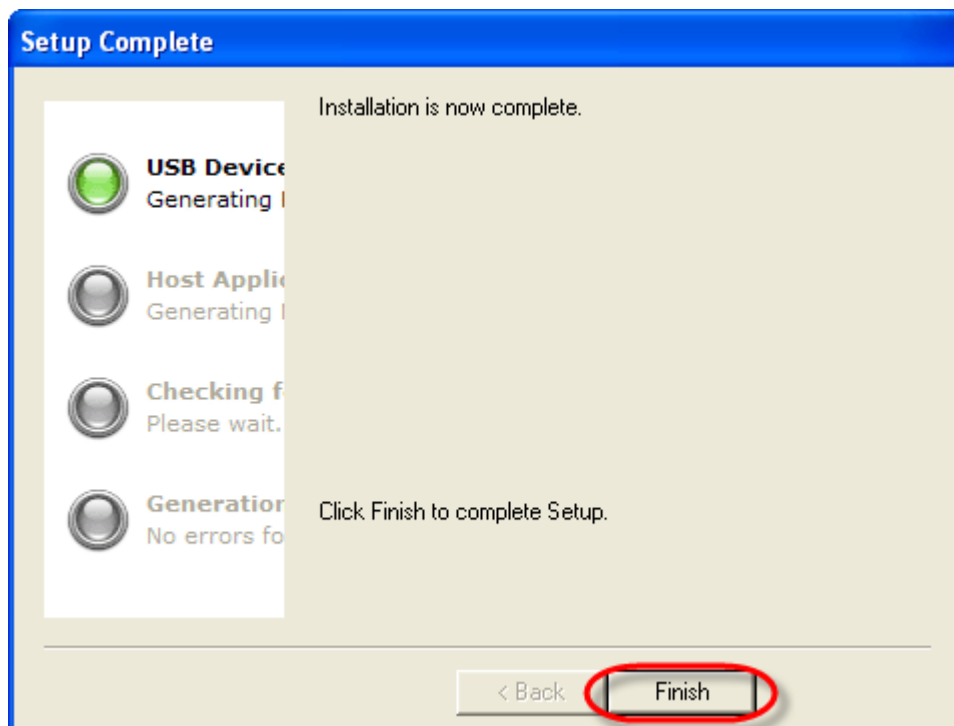
3. After user accepted the conditions and agreements, it starts installing the program instantly. In this case, program forces user to choose the folder location to install this program, it is more convenient to install the program according to the Default value; so, click **Next** as shown in the picture below;



4. Now, the program forces user to set Group position to create icon for calling this program; in this case, user should set the value according to the standard Default Value and then click **Next** as shown in the picture below;



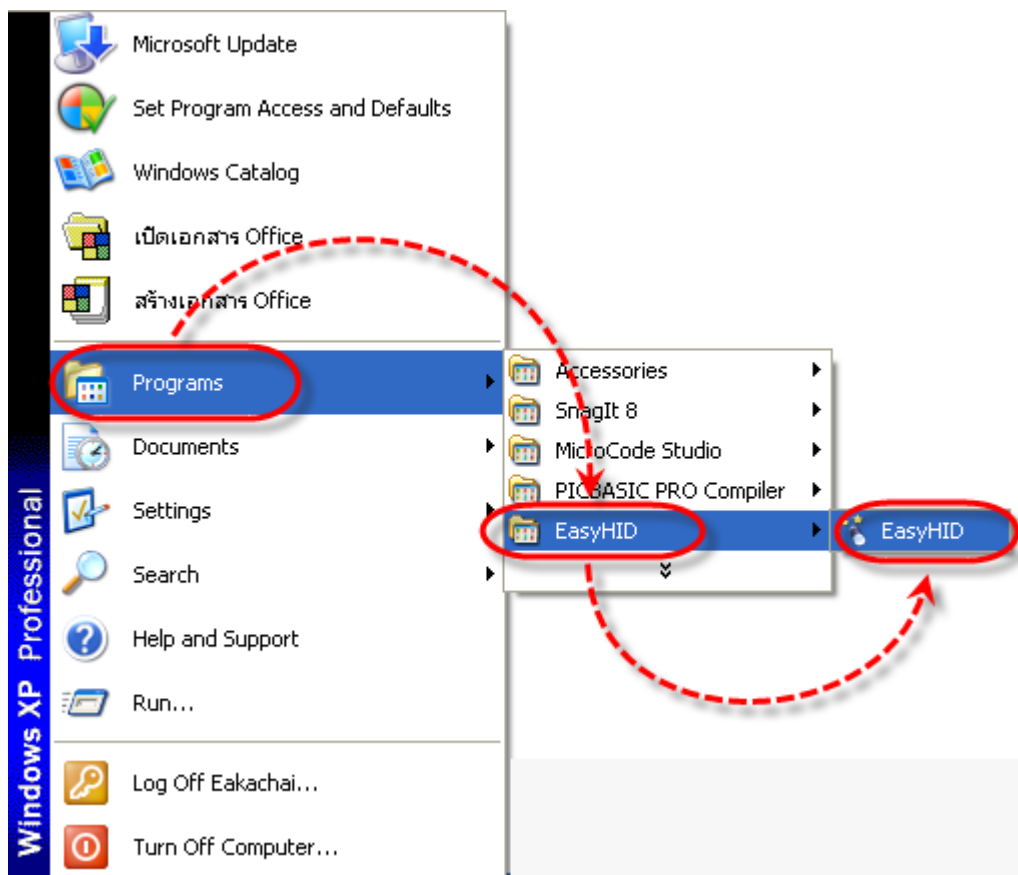
5. Now, the Program starts installing files into computer, please wait until the process of installing files is complete. Click **Finish** to finish the process of installing program as shown in the picture below;



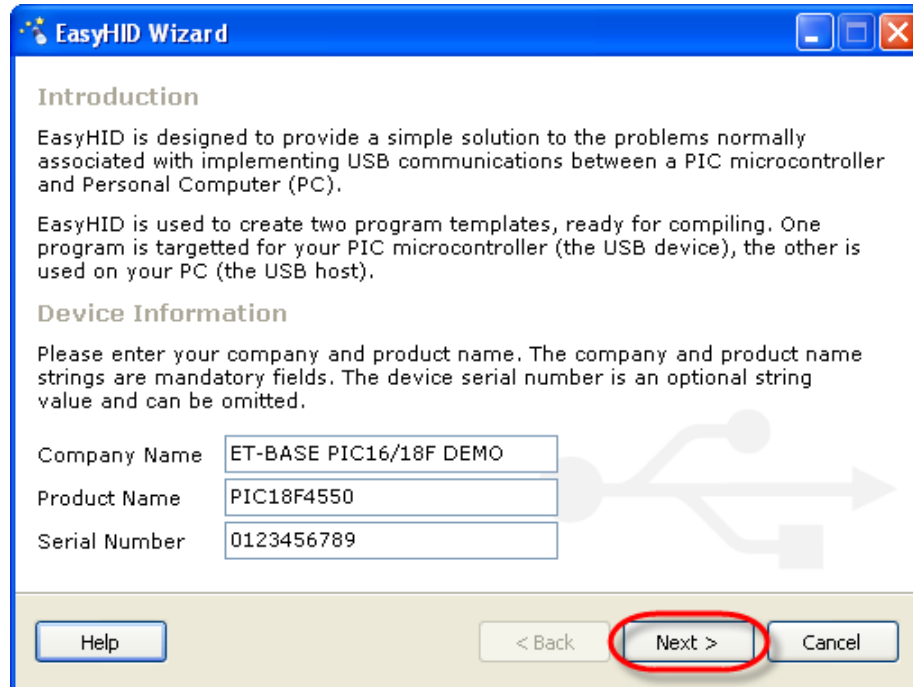
Example of using Program EasyHID to create Source Code

When user has installed the program completely; next, it calls and setup the program to create Source Code. In this case, it illustrates procedures for developing Program HID of Microcontroller No.PIC18F4550 by using the Hardware System of Board ET-BASE PIC16/18F that is compatible with is used with MCU No.PIC18F4550. It sets the function of USB to be HID to communicate and command Microcontroller through computer PC. It uses BASIC Language (Basic Pro compiler) to develop program on the side of Microcontroller and it uses Visual Basic Language to develop program on the side of computer PC. In this case, operation of this example program has been designed to ON/OFF LED on board by computer PC. Meanwhile, it reads value of pressing switch and reads Signal ANALOG from ADC to display on the screen of computer PC. These are procedures for using the program;

1. Call Program EasyHID; click **Menu Start → Program → EasyHID → EasyHID.**



- When program starts running, window EasyHID Wizard appears and user has to setup values; Company Name, Product Name, and Serial Number as required. Finally, click **Next** as shown in the example below.



EasyHID Wizard

Introduction

EasyHID is designed to provide a simple solution to the problems normally associated with implementing USB communications between a PIC microcontroller and Personal Computer (PC).

EasyHID is used to create two program templates, ready for compiling. One program is targetted for your PIC microcontroller (the USB device), the other is used on your PC (the USB host).

Device Information

Please enter your company and product name. The company and product name strings are mandatory fields. The device serial number is an optional string value and can be omitted.

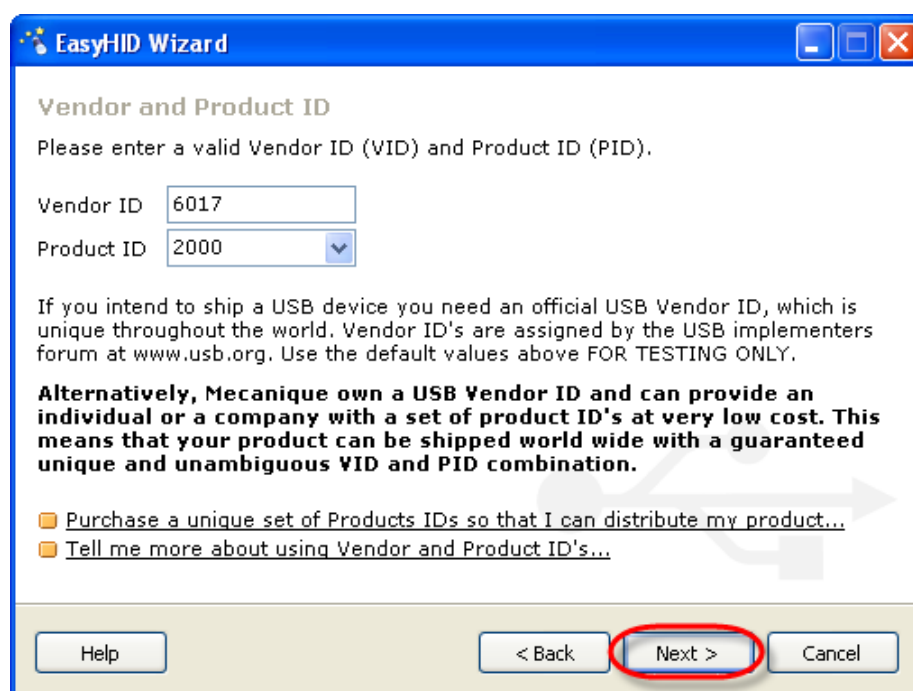
Company Name: ET-BASE PIC16/18F DEMO

Product Name: PIC18F4550

Serial Number: 0123456789

Buttons: Help, < Back, **Next >**, Cancel

- Now, Program forces user to setup **Vendor ID** and **Product ID** of USB device that user requires creating; in this case, user has to setup the values according to the Default Values of the program. Finally, click **Next** as shown in the picture below;



EasyHID Wizard

Vendor and Product ID

Please enter a valid Vendor ID (VID) and Product ID (PID).

Vendor ID: 6017

Product ID: 2000

If you intend to ship a USB device you need an official USB Vendor ID, which is unique throughout the world. Vendor ID's are assigned by the USB implementers forum at www.usb.org. Use the default values above FOR TESTING ONLY.

Alternatively, Mecanique own a USB Vendor ID and can provide an individual or a company with a set of product ID's at very low cost. This means that your product can be shipped world wide with a guaranteed unique and unambiguous VID and PID combination.

☐ Purchase a unique set of Products IDs so that I can distribute my product...

☐ Tell me more about using Vendor and Product ID's...

Buttons: Help, < Back, **Next >**, Cancel

4. Now, Program forces user to setup the communicating specifications of USB HID or Device that user will be created by Microcontroller PIC18F4550. These specifications are referred and used to create Source Code according to the following values;

- **Polling Input:** It is Time Cycle to ask for data from USB Device of USB Host; in this case, it is set as 10ms.
- **Polling Output:** It is Time Cycle to send data from USB Host to USB Device; in this case, it is set as 10ms.
- **Bus Power:** It is value of Current that USB Host will supply to USB Device through Connector USB; USB Host can supply 500mA Current at the maximum. In this case, it is set as 100mA Current and is multiplies by 2; so, it is set as 50.
- **Buffer Input and Buffer Output:** Normally, it can be set to be 64 Bytes at the maximum; in this case, it is set to be 3Byte according to the objectives. It initially designs the operational structure of HID as follows;
 - The first byte is contained the value of ADC (0... 255).
 - The second byte is contained the value of Switch (0 = SW Press, 1 = SW Release).
 - The third byte is contained the value of LED (0 = LED OFF, 1 = LED ON).

EasyHID Wizard

Configuration Details

The input polling interval is used by the host to request data from a USB device. The output polling interval is used by the host to send data to a USB device. Bus power is the maximum power consumption (x2) of the USB device on the bus.

Polling (Input) ms - host requests data from a USB device (max latency)

Polling (Output) ms - host sends data to a USB device (max latency)

Bus Power x2 mA

The input buffer (report) is sent by a USB device when requested to do so by the host. The output buffer (report) is sent by the host to a USB device.

Buffer (Input) bytes - USB device to host (64 bytes max)

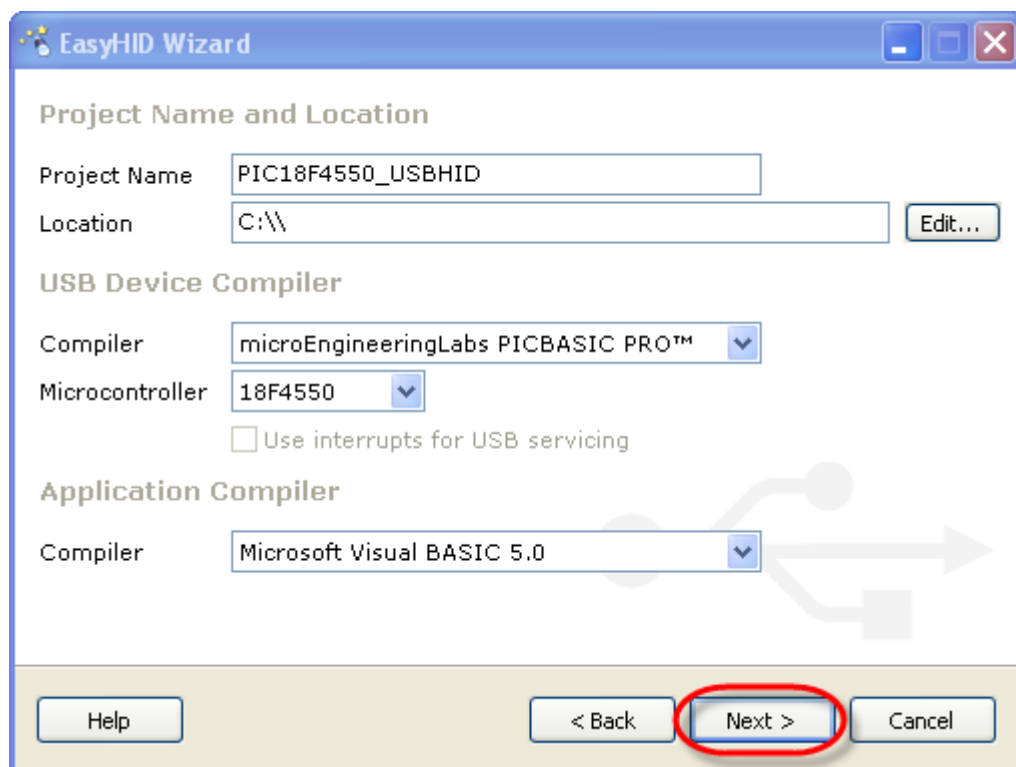
Buffer (Output) bytes - host to USB device (64 bytes max)

If you are unsure about the correct values to enter, use the recommended defaults.

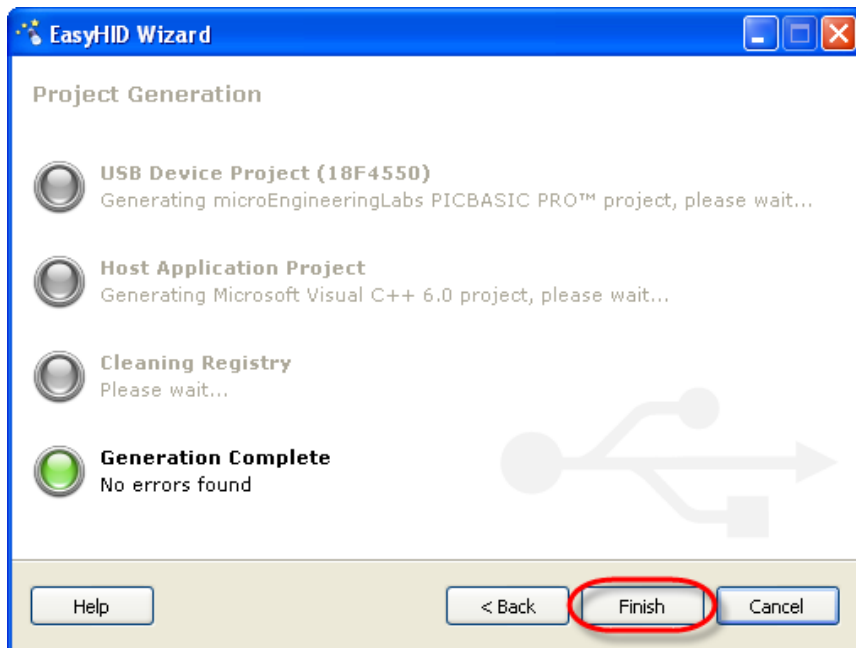
Help < Back **Next >** Cancel

5. Now, program forces user to setup **Project Name** and **Location** for storing the created file, including language format of Source Code that will be created by Program EasyHID. In this case, Program EasyHID creates Source Code in both parts of Microcontroller and program to command computer PC. It has to setup values as follows;

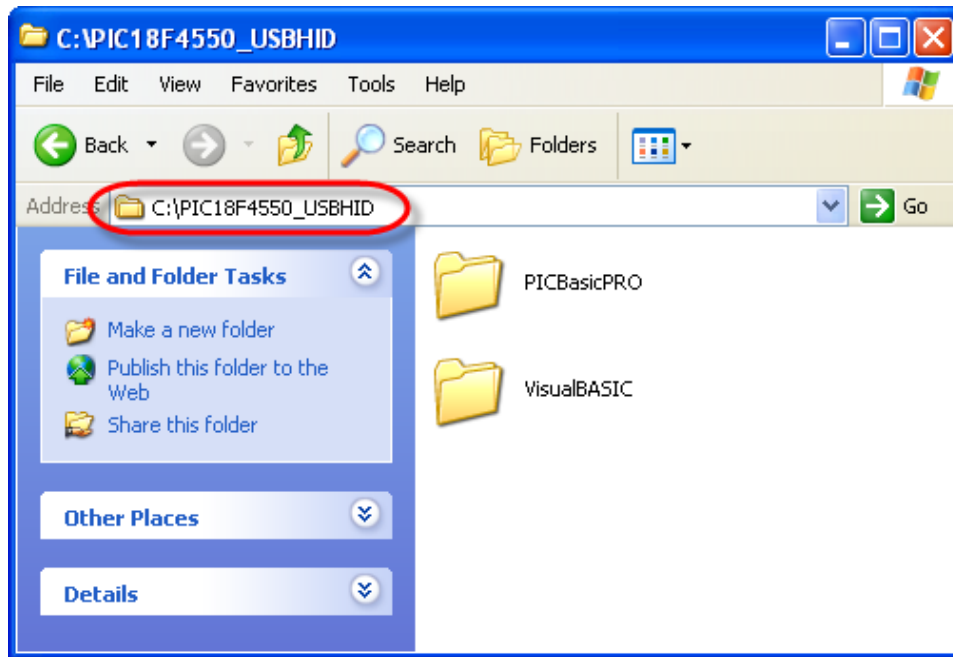
- Project Name and Location
 - Set **Project Name** as **PIC18F4550_USBHID**
 - Set **location** as **C:**
- Set **USB Device Compiler** as **BASIC Language (PIC Basic Pro Compiler)** for PIC18F4550; so, it has to set values as follows;
 - Set **Compiler** as **microEngineeringLabs PICBASIC PRO**
 - Set **Microcontroller** as **18F4550**
- Application Compiler
 - Set **Compiler** as **Microsoft Visual Basic 5.0**



6. When user has set specifications completely, Program starts creating Source Code instantly. In this case, user has to wait until the program shows the message "**Generation Complete**" and then click **Finish** as shown in the picture.



7. Refer to the previous step; if there is no any error, Program EasyHID creates Source Code completely according to the setting conditions. In this case, there are 2 sets of Source Code of program that have been created; it belongs to *PICBasicPRO* and *VisualBASIC* as shown in the example;



The part of Source Code that is BASIC Language (PIC Basic Pro Compiler), Program EasyHID creates Source Code, including the related File Library. The part of main File, it has created and provided variable and sub-function to connect with USB HID; in this case, there are 3 main functions as follows;

- **USBINIT:** This part initials the operation of MCU and USB Hardware internal MCU.
- **DoUSBIn:** It is function in the part of receiving data HID Packet from USB Host.
- **DoUSBOut:** it is function in the part of sending data through HID Packet to 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, transmit data
    RETURN
    
```

```

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
```

This is an example of Source Code that is BASIC Language from Program EasyHID.

The Source Code from Program EasyHID is incomplete because it only consists of basic functions for receiving-sending data between HID Device and USB Host; but there is no any part of checking conditions of data in Packets. So, user has to add more conditions into this part by self to setup the operation of program to run according to the objectives of user.

Refer to the operational format of USBHID Device in this example; it has set Microcontroller PIC18F4550 to detect and wait for receiving Packet HID from USB Host; then, it checks the detected data in the Packet to convert into condition for controlling ON/OFF LED. Next, it reads value of ADC and checks whether status of Push-Button Switch is pressed or released; and finally, it converts the value of ADC and Switch into HID Socket to send to USB Host.

It uses signals in the part of Hardware I/O of PIC18F4550 as follows;

- **RA0:** It is used to be Function ADC to read the value of voltage from the adjustable Volume. It sets the range of 8Bit ADC from 0 to 255.
- **RA1:** It is used to be Function Input Logic to read status of SW1 to check whether it is pressed or released. In this case, it converts value into data between 0 (pressed) and 1 (released).
- **RA2:** It is used to be Function Output Logic. It uses LED to display the LOGIC result; in the case, "0" is used for OFF LED and "1" is used for ON LED.

Refer to the operational conditions above, it has to set Packet HID of USB; in this case, the USB Buffer is divided into 2 sets. Firstly, it is used to receive data (*DoUSBIn*); and secondly, it is used to send data (*DoUSBOut*). Moreover, each USB Buffer has the same size of 3 Byte. User should set the structure of data for each byte in Packet HID as follows;

- The first byte contains the value of ADC that ranges from 0 to 255.
- The second byte contains the status of pressing Push Button Switch that is in the range of 0 and 1. In this case, "0" means that Switch is pressed; on the other hand, "1" means that Switch is released.
- The third byte contains the Control Code for ON/OFF LED that is in the range of 0 and 1. In this case, "0" is used for OFF LED while as "1" is used for ON LED.

On the side of sending Packet HID, user has to set Program to read value of ADC and SW1; next, it converts the value into data and contains it in the first byte and the second byte of USB Buffer, respectively. This Packet of sending data ignores value of LED ON/OFF.

On the side of receiving Packet HID, it ignores value of data in the first byte and the second byte. In this case, it only cares the value of data in the third byte; it checks whether it is 0 or 1 and then it commands to ON/OFF LED according to the receiving data from Packet HID.

User has to add more commands of program to setup the operation of Pin RA0(ADC), RA1(Input Logic, value of SW1), and RA2 (Output Logic, drive LED), including to the part of checking data in Packet HID as shown in the example below;

```

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

```

This example shows the BASIC LANGUAGE Code that has been edited and added completely.

In case of using BASIC Language (PIC Basic Pro Compiler), there is a part of CODE that is written by BASIC Language; moreover, user has to setup Configuration values to choose the operation for MCU through File "C:\PBP\18F4550.INC".

```

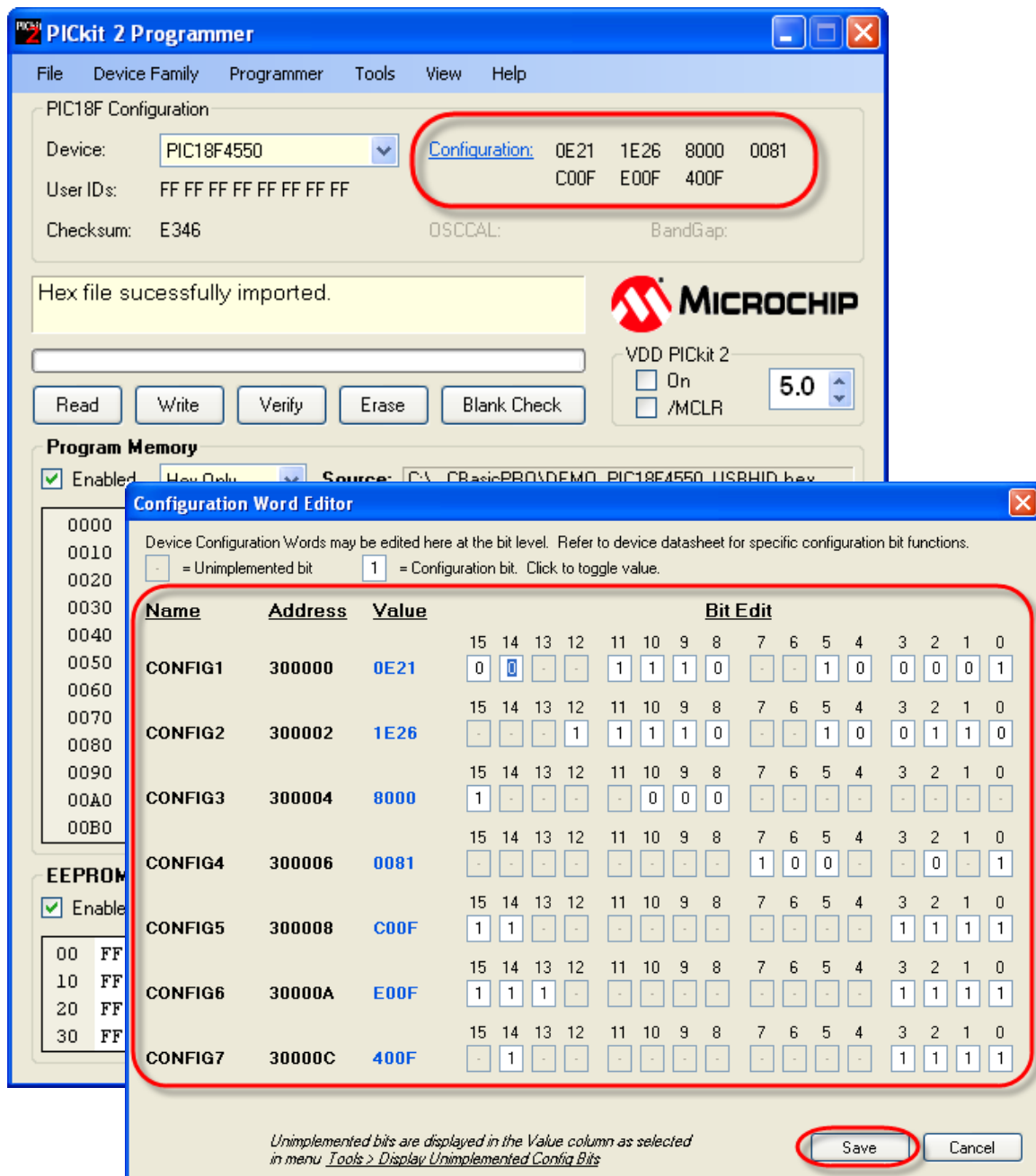
'ET-BASE PIC16/18F Hardware used X-TAL=8MHz
'Setup Clock Config -> 8MHz/2=4MHz -> 4MHzxPLL(96MHz)->96MHz/2=48MHz
'Config1 = 0x0E21 = 0000 1110 : 0010 0001
'Config2 = 0x1E26 = 0001 1110 : 0010 0110
'Config3 = 0x8000 = 1000 0000 : 0000 0000
'Config4 = 0x0081 = 0000 0000 : 1000 0001
'Config5 = 0xC00F = 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

```

This example shows how to setup Configuration Bit for PIC18F4550 of Board ET-BASE PIC16/18F.

In case of using PicKit2 to be programmer device, user can check the Configuration value from the program in the step of programming. If user has set the Configuration value in File "C:\PBP\18F4550.INC" correctly, the Configuration value on the window program will be changed according to the setting value instantly after loaded HEX into program. If it does not match or user forgets to set the Configuration value in Source Code, user can edit the new Configuration value in the window program of PICKit2 by self as shown in the example;



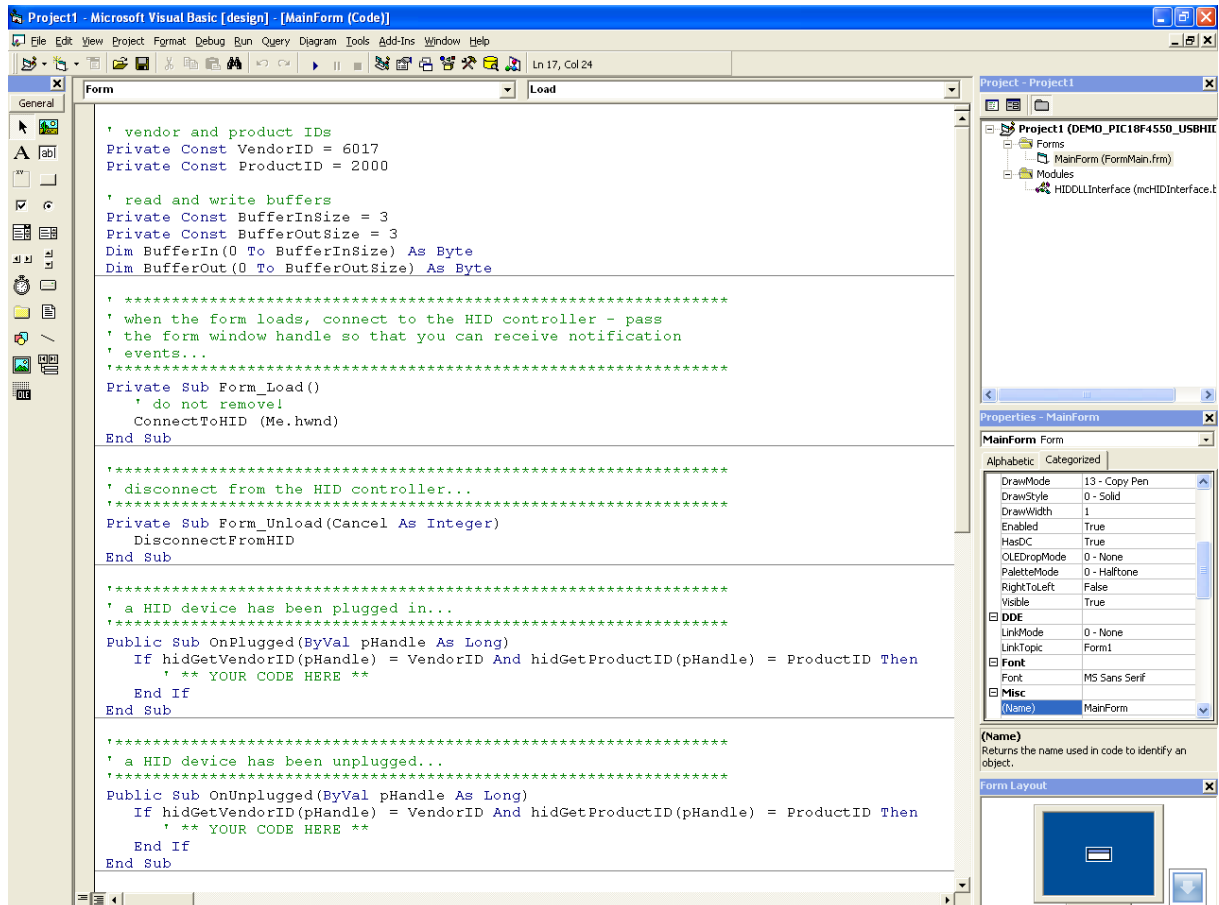
Window shows feature of the Configuration values of PIC18F4550 of Program PICKit2.

How to develop Source Code of Visual Basic

The Source Code of Visual Basic is the same as the Source Code that has been created by Program EasyHID; there is only the main structure of program to communicate with Microcontroller through USB HID. In this case, it consists of declaring necessary variable and creating function to initial and send/receive data. User has to write program to setup more

operational conditions according to the objectives by self; in this case, user cannot use the Source Code directly.

User has to create Form to arrange Component for communicating with user and write program to link conditions of operation and display together by self. This Source Code is just a tool that is used to communicate, send-receive data of USB HID between USB Host of computer PC and Microcontroller only.



```

' 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 microcontroller...
    End If
End Sub

```

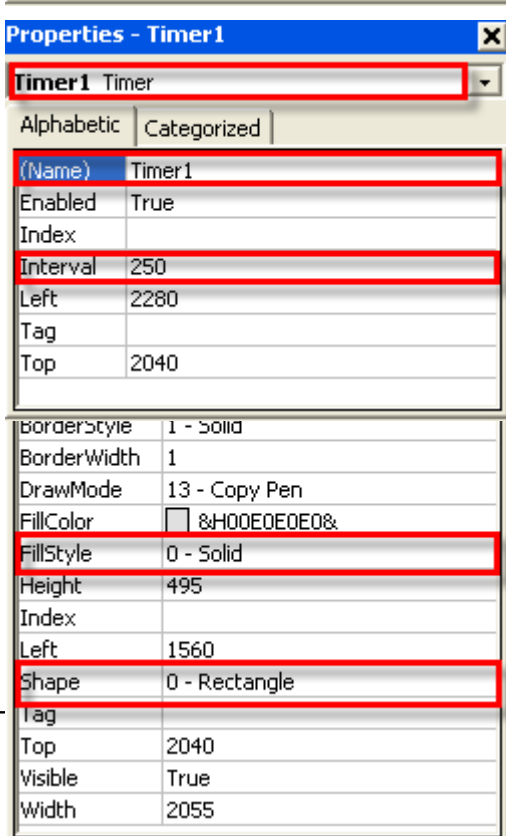
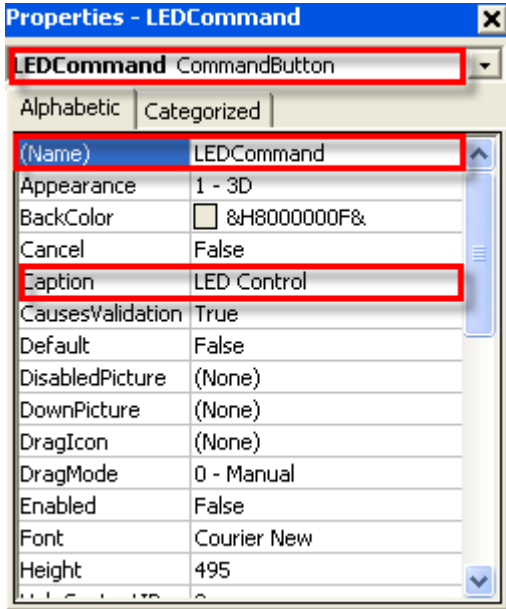
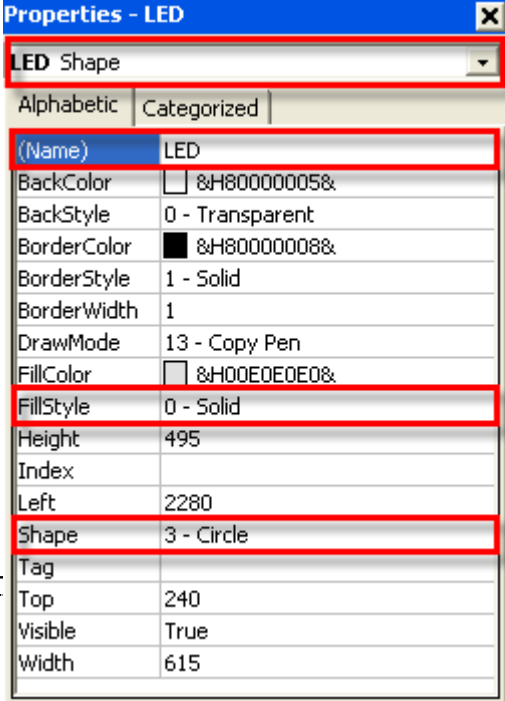
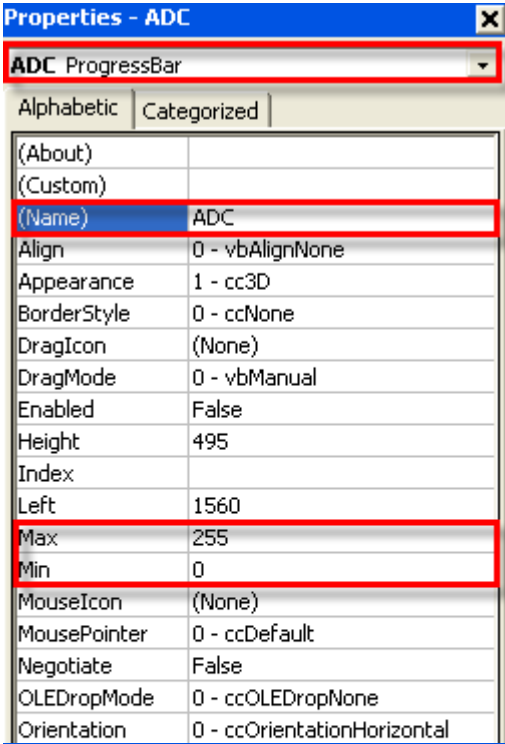
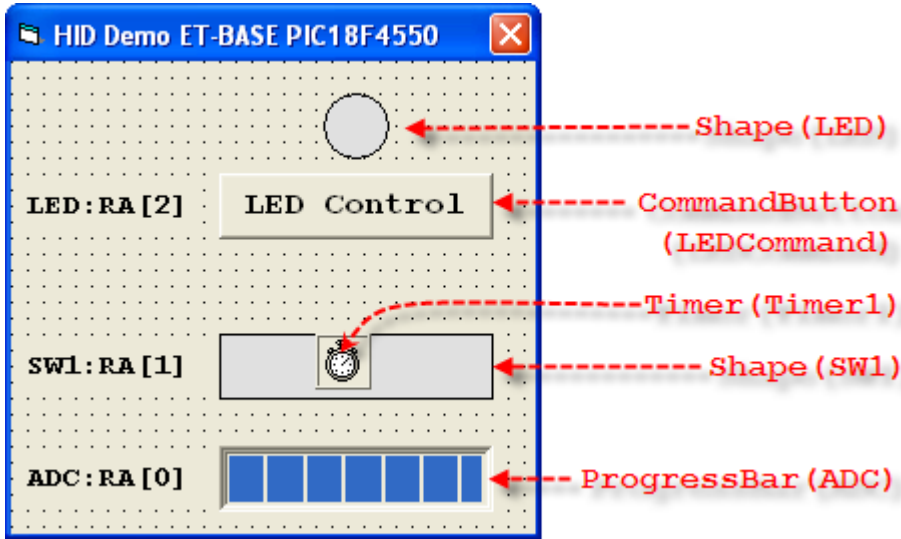
```
End If
End Sub

'*****
' 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
```

This example shows Source Code of Visual Basic from Program EasyHID.

Refer to the Source Code above, open Form that is a blank form has not been arranged any Component yet; in this case, user should arrange and adjust the Component as shown in the example.



This shows how to set and adjust Component on Form of Visual Basic.

After user has arranged Component on Form and adjusted value of Component completely, user has to write Code to link conditions of the operation together. Refer to the format in the example; it uses *CommandButton* called *LEDCommand* to be button to send Command ON/OFF LED on board. It operates in the format of toggle; when user clicks the button, it reverses the status of LED to be opposite of the current status. For example, if it is in status OFF, it is reversed to the status ON; on the other hand, if it is in status ON, it is reversed to the status OFF. The value of data for Command ON/OFF LED is contained in the third Buffer of Packet HID.

It uses Timer to be timing to send Status Refresh to Microcontroller and then Microcontroller will echo with the status of Switch SW1 and value of ADC. When the Microcontroller has echoed with data of Packet HID, program detects and responds to the operation by function **OnRead** in **Public Sub OnRead(ByVal pHandle As Long);** in this case, user has to write program to setup conditions of the operation in this function. It reads value of data in the Packet; it converts the value in the first order into ADC for *ProgressBar*; and it checks the value in the second order to display the status of Switch SW1 by *Shape(SW1)*. In this case, it displays the operation by colors to notify user to know SW is pressed or released.

<code>Const InputON = &HFF00&</code>	<code>' Green Color</code>
<code>Const InputOFF = &H80000004</code>	<code>' Menu Color</code>
<code>Const OutputON = &HFF&</code>	<code>' Red Color</code>
<code>Const OutputOFF = &HFFFFFF8</code>	<code>' Menu Color</code>

```

' 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 microcontroller...
        ' ** 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 byte 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

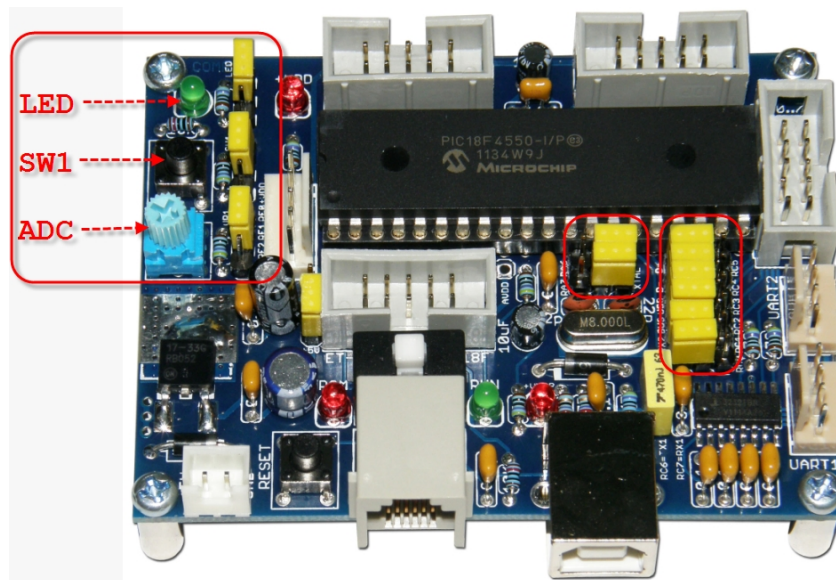
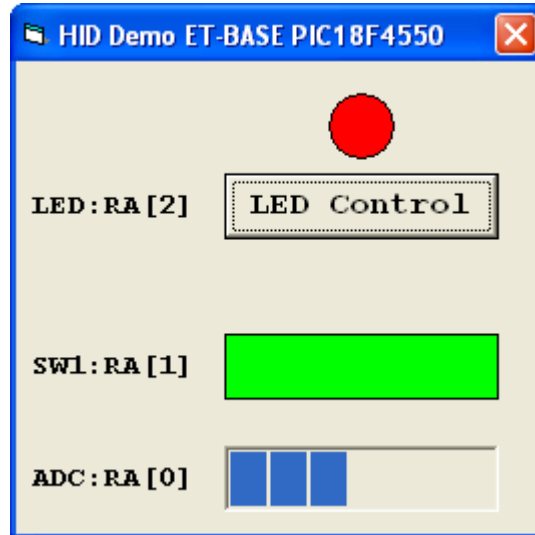
    ' write the data (don't forget, pass the whole array)...
    hidWriteEx VendorID, ProductID, BufferOut(0)

End Sub

```

This example shows Source Code of Visual basic that has been edited and changed completely.

When user compiles and run the program, it shows the operating result as shown in the picture below;



When user requires testing the operation of program, it has to interface USB Cable to connect Board Microcontroller "ET-BASE PIC16/18F(PIC18F4550) that has been programmed HEX File from Basic PRO Compiler completely; next, user can control and command LED on board through this window program instantly. When user pressed SW1 on board, user can see the status of SW1 on the window program has been changed according to pressing and releasing the Switch. In the meanwhile, when user has adjusted the Resistor, the *ProgressBar* that displays the value of ADC has been changed as well.