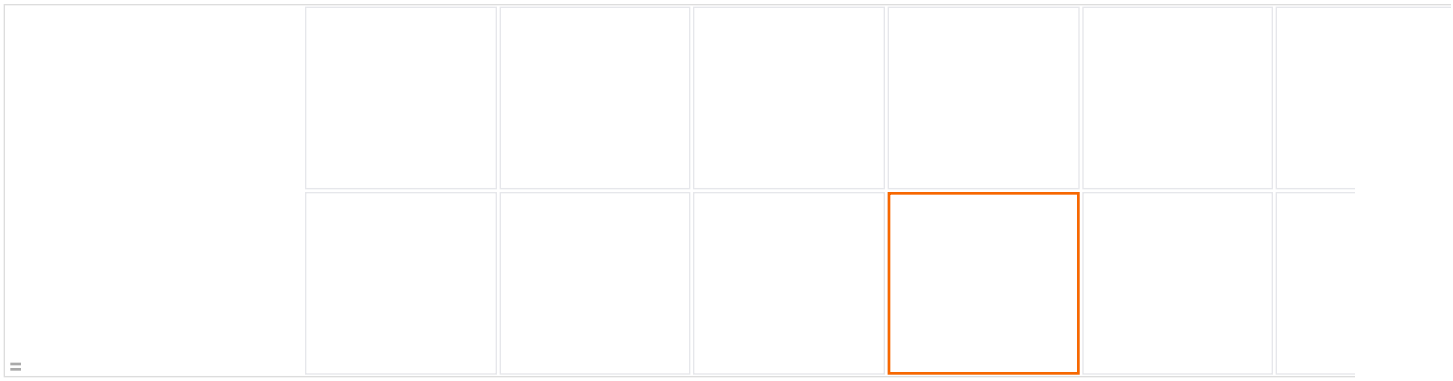
EMBEDDED LABORATORY

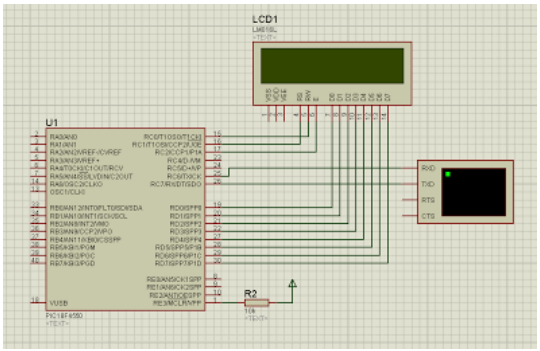


USB HID Communication using PIC (Part-1)

by Unknown *on* August 27, 2016 *in* HID, Microchip, PIC, PIC18f4550, USB

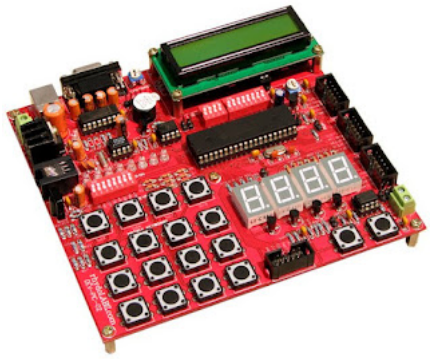
PIC Micro-controllers are one of the most popular from the Microchip. In this post i will describe you guys how to use PIC18F4550 micro-controller for developing a USB based product.
This tutorial is divided into two parts, in the first part i will demonstrate how to use mikroC compiler for developing a simple USB HID project and in the another part i will show you how to use CCS PIC-C compiler for developing simple USB HID Project. To read Part-2 click here.

Compiler Used: mikroC 5.30
This demo is tested on Labcenter Electronics Proteus and PIC18F4550 USB Development Board.



Schematic Diagram

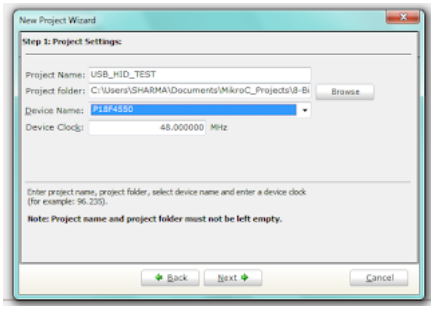
One can use Proteus for this or any development board of your choice, like the one i am using.



Development Board

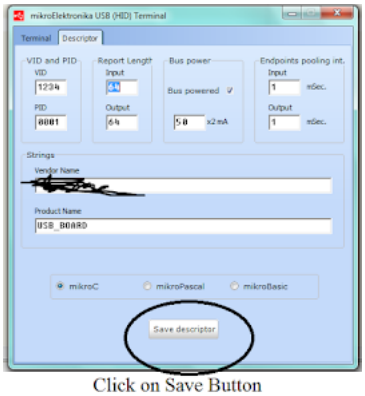
USB is a much complex device to handle but mikroC has inbuilt function which made this task extremely simple for all the beginners.
Create a new project in mikroC and select the mico-controller PIC18F4550 with Device Clock value of 48MHz (Though i am using 20MHz Crystal Oscillator but by enabling the PLL of PIC, the device clock is boosted to 48MHz)

Those who still face the difficulty in making project in mikroC, can watch this video.



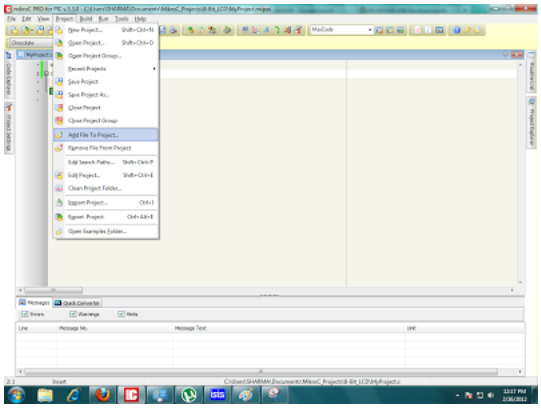
Project Wizard

Now time to write Code for USB HID Class but before doing so, you have to provide the VendorID, Product ID etc. This can be done by following the simple instructions in MikroC. Goto Tools -> HID Terminal and set these values as shown in the following picture.



HID Class Configuration

After following the above step a new file is generated, and we have to add this file into our project, which is shown below.



Add Descriptor File to Project

Now copy and paste the following code, the code is self explanatory as comments are provided.

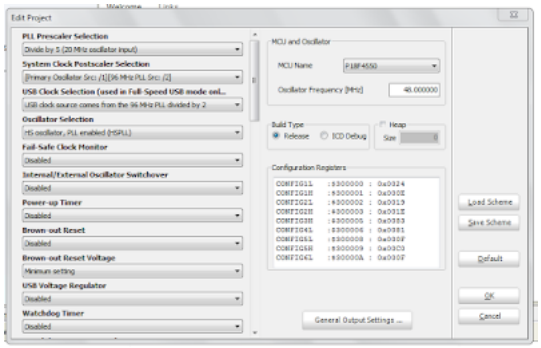
Code

```
unsigned char Read_Buffer[16] absolute 0x500;
unsigned char Write_Buffer[16]absolute 0x510;
unsigned char num,flag;
void interrupt()
{
 USB_Interrupt_Proc();
 TMR0L = 100;       //Reload Value
 INTCON.TMR0IF = 0;    //Re-Enable Timer-0 Interrupt
}
//LCD 8-bit Mode Connection
sbit LCD8_RS at RC1_bit;
sbit LCD8_RW at RC0_bit;
sbit LCD8_EN at RC2_bit;
sbit LCD8_D7 at RD7_bit;
sbit LCD8_D6 at RD6_bit;
sbit LCD8_D5 at RD5_bit;
sbit LCD8_D4 at RD4_bit;
sbit LCD8_D3 at RD3_bit;
sbit LCD8_D2 at RD2_bit;
sbit LCD8_D1 at RD1_bit;
sbit LCD8_D0 at RD0_bit;
sbit LCD8_RS_Direction at TRISC1_bit;
sbit LCD8_RW_Direction at TRISC0_bit;
sbit LCD8_EN_Direction at TRISC2_bit;
sbit LCD8_D7_Direction at TRISD7_bit;
sbit LCD8_D6_Direction at TRISD6_bit;
sbit LCD8_D5_Direction at TRISD5_bit;
sbit LCD8_D4_Direction at TRISD4_bit;
sbit LCD8_D3_Direction at TRISD3_bit;
sbit LCD8_D2_Direction at TRISD2_bit;
sbit LCD8_D1_Direction at TRISD1_bit;
sbit LCD8_D0_Direction at TRISD0_bit;
// End Lcd8 module connections
char i;               // Loop variable
void UART1_Write_Text_Newline(unsigned char msg[])
{
 UART1_Write_Text(msg);
 UART1_Write(10);
 UART1_Write(13);
}
void clear_buffer(unsigned char buffer[])
{
 unsigned int i = 0;
 while(buffer[i] != '\0')
 {
 buffer[i] = '\0';
 i++;
 }
}
//
void main()
```

```
{
 UART1_Init(9600);
 Delay_ms(100);
 UART1_Write_Text("USB Test Program");
 ADCON1 |= 0x0F;          // Configure AN pins as digital
 CMCON |= 7;             // Disable comparators
 TRISB = 0x00;
 TRISC = 0x80;
 Lcd8_Init();            // Initialize Lcd8
 Delay_ms(100);
 Lcd8_Cmd(_LCD_CLEAR);       // Clear display
 Delay_ms(100);
 Lcd8_Cmd(_LCD_CURSOR_OFF);     // Cursor off
 Delay_ms(100);
 Lcd8_Out(1,3,"PIC18F4550");        // Write text in first row
 Delay_ms(100);
 Lcd8_Out(2,3,"USB Example!");        // Write text in second row
 Delay_ms(2000);
 INTCON = 0;
 INTCON2 = 0xF5;
 INTCON3 = 0xC0;
 RCON.IPEN = 0;
 PIE1 = 0;
 PIE2 = 0;
 PIR1 = 0;
 PIR2 = 0;
//
// Configure TIMER 0 for 3.3ms interrupts. Set prescaler to 256
// and load TMR0L to 100 so that the time interval for timer
// interrupts at 48MHz is 256.(256-100).0.083 = 3.3ms
//
// The timer is in 8-bit mode by default
 T0CON = 0x47; // Prescaler = 256
 TMR0L = 100; // Timer count is 256-156 = 100
 INTCON.TMR0IE = 1; // Enable T0IE
 T0CON.TMR0ON = 1; // Turn Timer 0 ON
 INTCON = 0xE0; // Enable interrupts
 //
 // Enable USB port
 //
 UART1_Write(10);
 UART1_Write(13);
 UART1_Write_Text_Newline("Data is Ready to be Received from the PC");
 Hid_Enable(&Read_Buffer,&Write_Buffer);
 Delay_ms(2000);
 // Read from the USB port. Number of bytes read is in num
 start:
 while(Hid_Read() == 0);  //Stay Here if Data is Not Coming from Serial Port
 //If Some Data is Coming then move forward and check whether the keyword start is coming or not
 if(strncmp(Read_Buffer,"S",1) == 0)
 {
 Lcd8_Cmd(_LCD_CLEAR);
 Lcd8_Out(1,2,"Authentication");
 Lcd8_Out(2,8,"OK");
 goto loop;
 }
 else
 {
 Lcd8_Cmd(_LCD_CLEAR);
 Lcd8_Out(1,2,"Authentication");
 Lcd8_Out(2,5,"Fails!");
 goto start;
 }
 loop:
//Now Authentication is Successfull Lets Try Something else
//Lets Display the Data Coming from the USB HID Port to the LCD
 Delay_ms(1000);
 Lcd8_Cmd(_LCD_CLEAR);
 Lcd8_Out(1,1,"Received Data:-");
 flag = 0;
 loop_second:
 clear_buffer(Read_Buffer);
 while(Hid_Read() == 0)
```

```
 {
 if(flag == 0)
 {
  Lcd8_Out(2,1,"No Data");
  flag = 1;
 }
 }
 Lcd8_Cmd(_LCD_CLEAR);
 Lcd8_Out(1,1,"Received Data:-");
 Lcd8_Out(2,1,Read_Buffer);
 goto loop_second;
 Delay_ms(1000);
 Hid_Disable();
 Lcd8_Out(1,1,"HID DISABLE");
}
```

Now time to edit the the Configuration bit, as they play a crucial part while dealing with the PIC Micro's.

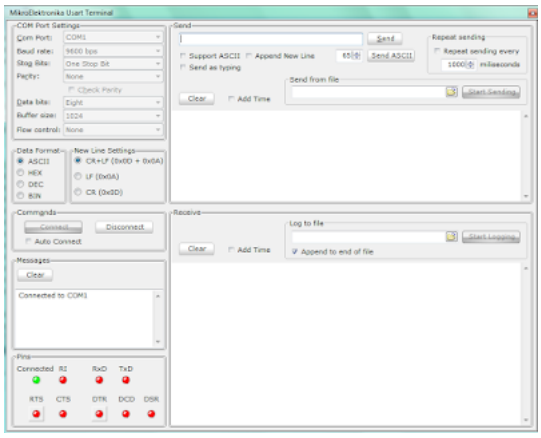


Configuration Editor

Edit these configuration bits and now build the project. After building the project hex file is generated which needs to be programmed in the micro-controller using your programmer/debugger.

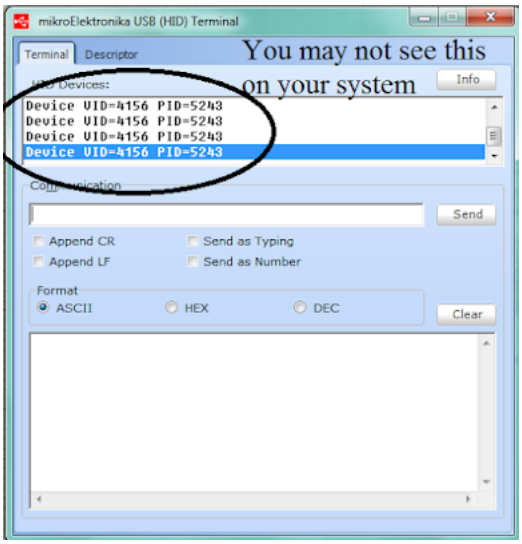
Now we are in a position to test the USB HID Connection.

Before connecting the board to USB port or running the simulation in Proteus, open Hid Termal and USART terminal present in the mikroC tools menu.
Configure the USART Terminal to 9600 Baud Rate and Select your COM port i am having COM1 on my system. Click on Connect Button, You will get the status on the USART Terminal.



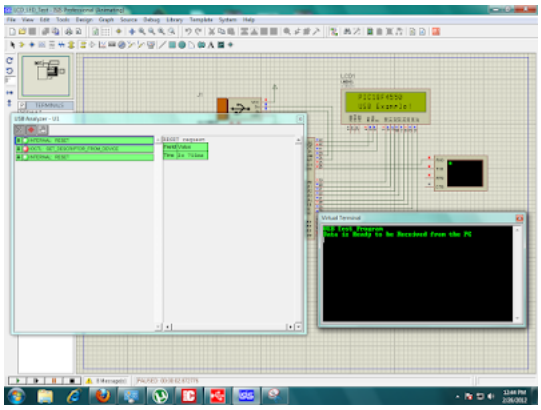
USART Terminal of mikroC

Now open the HID Terminal of mikroC, this software will list all the USB HID devices connected to the PC, as shown below.

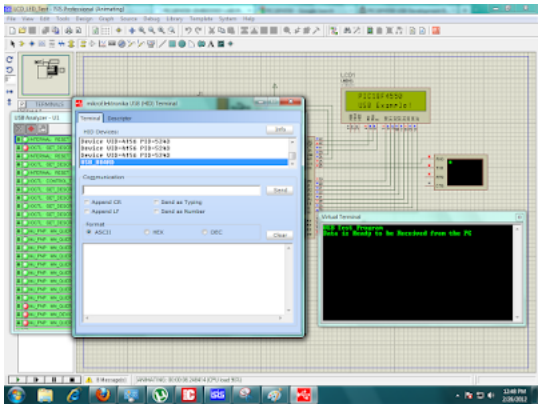


USB HID Terminal of mikroC

Now connect your hardware with PC, I am showing the simulation which i run on the Simulation Software and after this i will show the working of project on real development board.

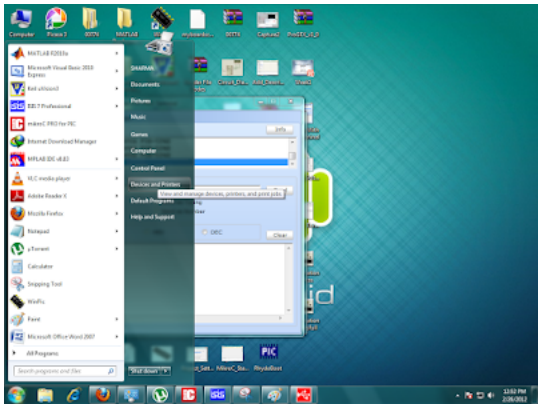


USB Enumeration is Started

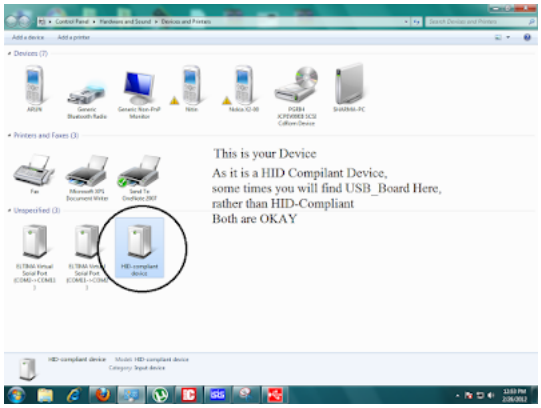


Hurray! our devices is detected by mikroC HID Terminal

Let's check whether we can see the same thing in Windows Device Manager.



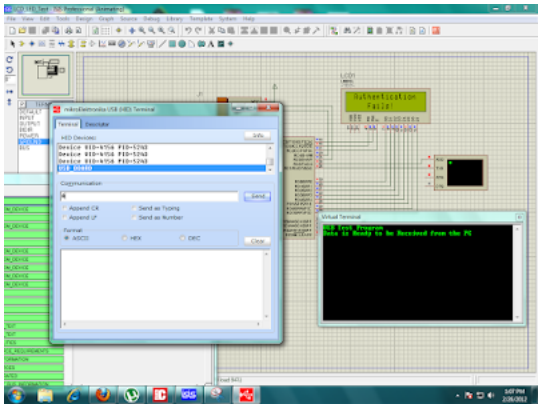
Open Windows Device Manager



That's Great Our Device is Detected in Windows 7

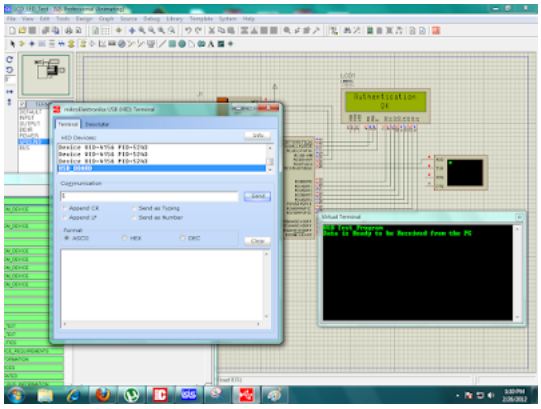
Now its time to send some data from PC to PIC18F4550. First i have to send 'S' through USB, this is just like a security check, if one doesn't send this character then we will continuously get Authentication Fail.
In order to proceed properly you have to send 'S' first then proceed further.
The firmware is written in such a way, that the string sent by Hid-Terminal will be displayed on LCD.

First see if one send wrong character, like i am sending 'A' from HID-Terminal.



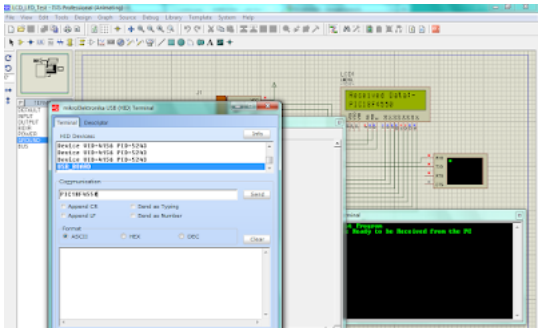
Authentication Failed

Now send the correct code from HID Terminal which is 'S'.

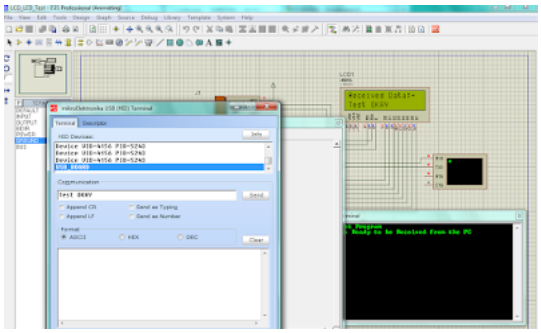


Authentication Passed

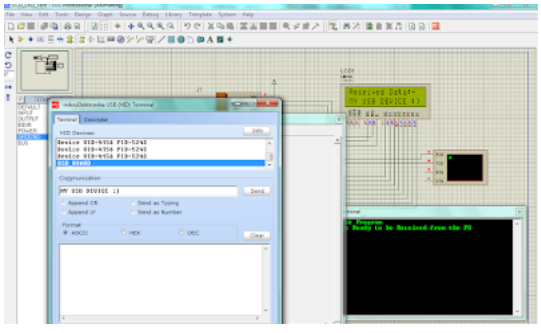
Now we can test, whatever we send from HID Terminal will be displayed on LCD.



Test 1



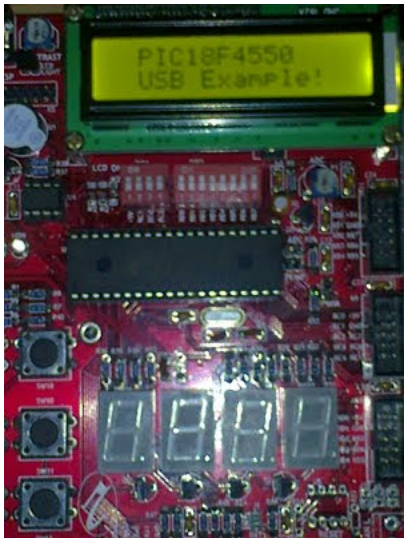
Test 2



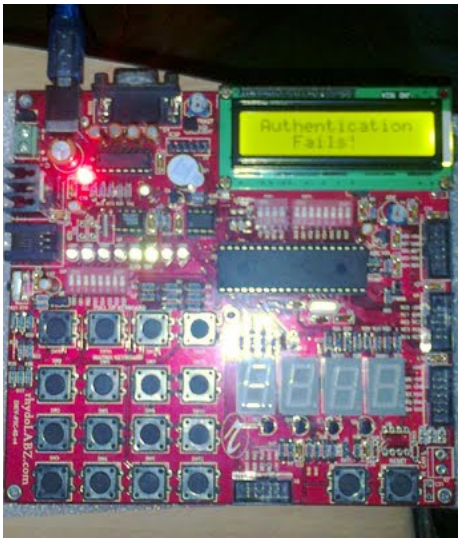
Test 3

Hey guys, finally you are done with your first USB HID Project.

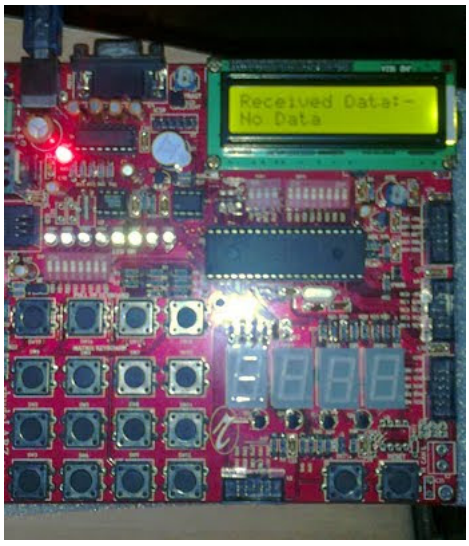
But as i promised, i will show the same demo for my Development board which are present below.



Board Connected to PC



Authentication Failed



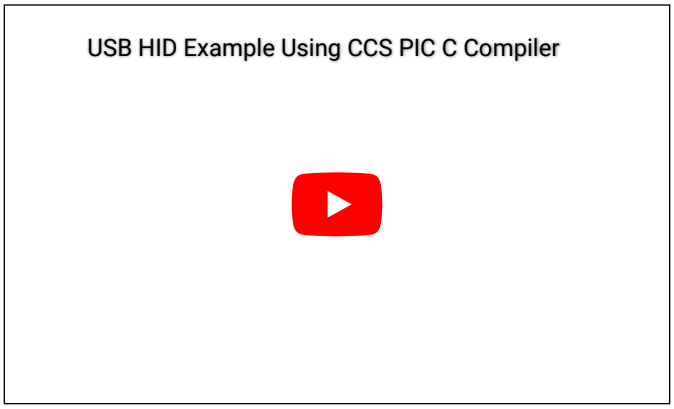
Authentication Passed

Test Data

Update: We had made a Software similar to HID Terminal provided my mikroC in Visual Basic, which uses mcHID.dll to communicate over USB HID Devices.

Embedded Laboratory USB HID Software

The following video demonstrate its usage.



To download the application click here.

Tags # HID # Microchip

About Unknown

Embedded Laboratory is a group of Electronics Hobbyist and Enthusiast, working on, to provide quality content to the guys starting in the Embedded Domain.

RELATED POSTS:

USB HID Communication using PIC (Part-2)

USB HID Communication using PIC (Part-1)

2 comments:

Unknown
July 31, 2018 at 12:24AM

how to upload hex file using micro C?

Reply

Unknown
August 25, 2018 at 10:14AM

hello dear brother
can you help me
I am going to design a project using C# application. and I want to use USB_HID PIC18F4550 as a dongle license for my C# application.
for more details, I need to send string word from USB_HID to my PC and then read it by C# application. This string word I will use it as Key a license for my application.
Thanks in advance

Reply

To leave a comment, click the button below to sign in with Blogger.

SIGN IN WITH BLOGGER

RECENT POST

- Building a Weather Station Web Server with OTA Updates Using ESP32 and ESP-IDF Framework
  Embedded Laboratory Aug 26, 2023
- Accelerometer Data Visualization on Desktop and Android using Qt
  Embedded Laboratory Aug 04, 2023
- Sensor Less Weather Station using ESP32 Open Weather Map Website and LVGL
  Embedded Laboratory Jul 13, 2023
- Temperature Controlled Switch without Microcontroller
  Embedded Laboratory May 28, 2023

ARDUINO POST

- Accelerometer Data Visualization on Desktop and Android using Qt
  Embedded Laboratory Aug 04, 2023
- Temperature and Humidity Graph Using ESP32 ILI9341 DHT11 and LVGL
  Embedded Laboratory Jan 03, 2023
- Weather Station using Arduino and ESP8266 with Open Weather API using AT Command
  Embedded Laboratory Feb 20, 2021
- Displaying Images on 3.2 Inch TFT using Arduino
  Embedded Laboratory Apr 29, 2018

POPULAR

- Serial Communication in Raspberry Pi Using Python
- Serial Communication Using Python
- OLED I2C Display Using Microchip PIC Microcontroller
- Posting DHT11 Values to ThingSpeak Using Nodemcu (Tutorial-5)