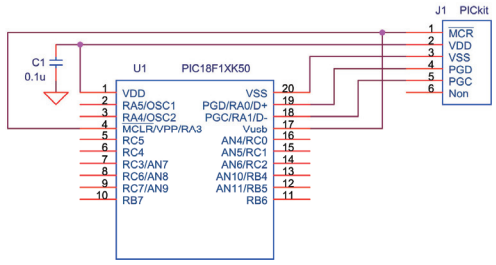
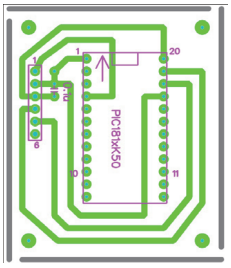
【回路】

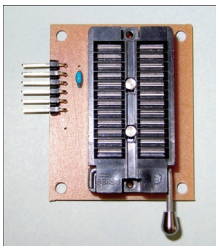
PIC18F40K15　書込みアダプタ

回路図



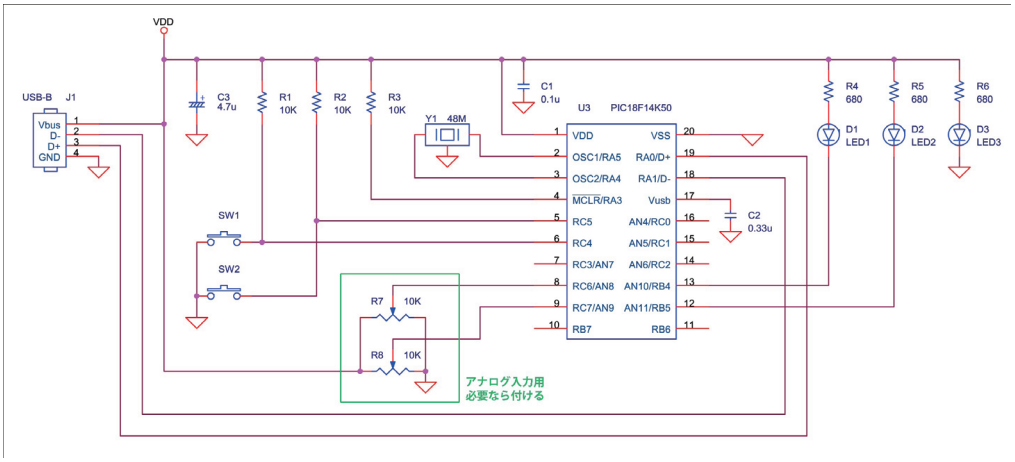
部品面





PIC18F40K15　USBテスト回路

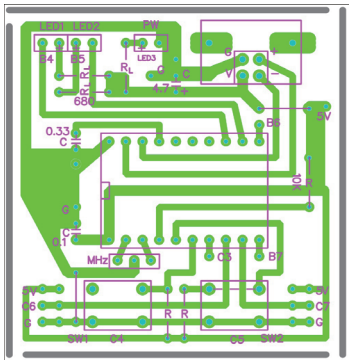
回路図



アナログ入力の可変抵抗は、今後のテストプログラムで使用するかも

記号	名称	個数
USB-B	USBコネクタBタイプ	1
C1	セラミックコンデンサ　0.1μF	1
C2	セラミックコンデンサ　0.33μF	1
C3	電界コンデンサ　4.7μF～10μF	1
R1～R2	抵抗　5KΩ～10KΩ	2
R3	抵抗　10KΩ	1
R4～R6	抵抗　680Ω～1KΩ	3
R7～R8	可変抵抗　5KΩ～10KΩ	2
D1～D3	LED	3
Y1	セラミック発振子 コンデンサ内蔵タイプ、48MHz	1
SW1～SW2	押しボタンスイッチ 押したときだけON	2
U3	ゼロプレッシャー（ZIF）ICソケット	1
	ICソケット	1
	PIC18F14K50	1

部品面

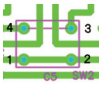


PICのICソケットには、ZIFソケットを使用

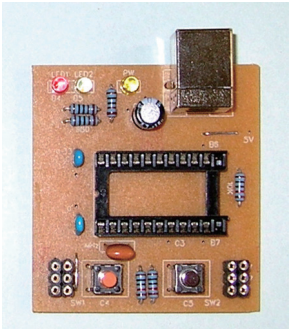
RL：680Ω～1KΩ程度
LEDの電流制限

R：プルアップ
10KΩ程度
（スイッチの10Kはもう少し低くても良い）

SWは内部で、1－2、3－4　が接続されている



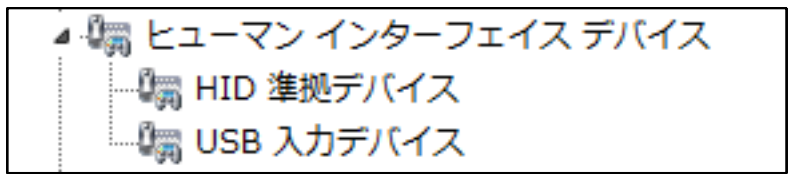
注意：RC4とRC5を略して、C4とC5と記している。



24ピンのZIFを使用＆ZIFが抵抗に当たらないように、
ZIFはICソケットに挿すようにした。

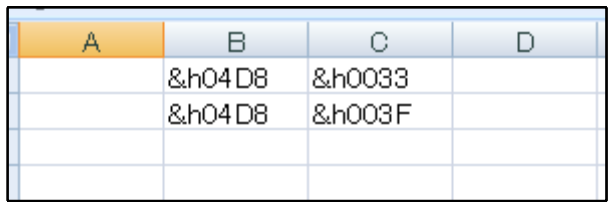
【動作確認】

1）テストボードをパソコンに接続
[デバイスマネージャー]を開いて、[ヒューマンインターフェイスデバイス]の中に、
「USB入力デバイス」が追加されているのを確認する。



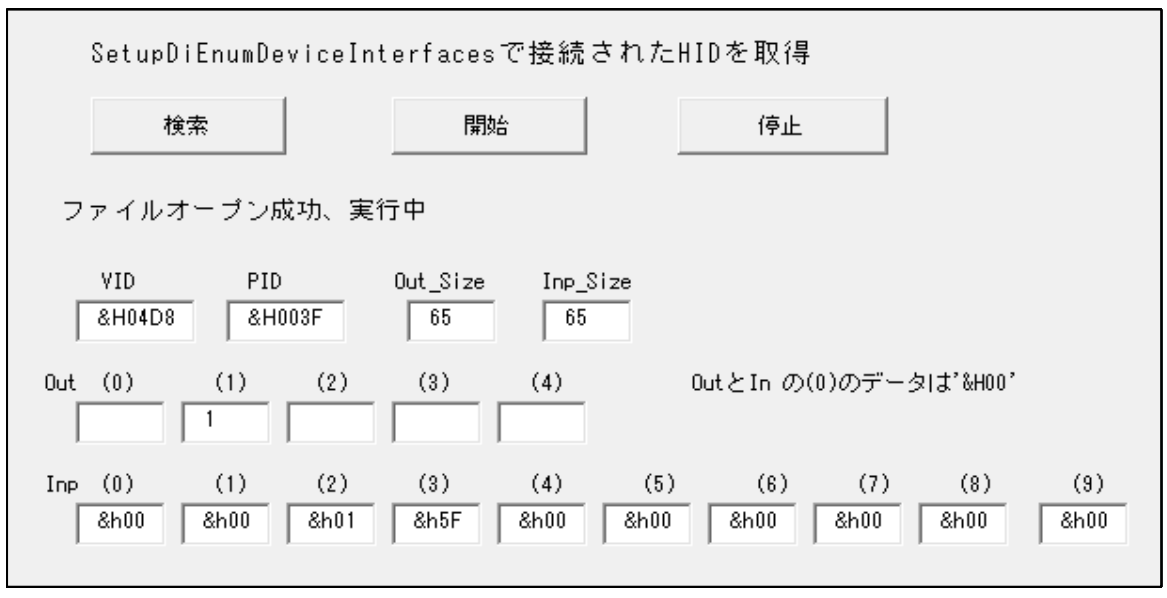
2）テストプログラムの「HID_TEST.xls」を起動

3）[検索]ボタン
HID接続のUSB機器のベンダーID（VID)とデバイスID（PID）の一覧をエクセルのセルに表示



A	B	C	D
	&h04D8	&h0033	
	&h04D8	&h003F	

4）VID と PID 値を確認
0x04D8　　// Vendor ID
0x003F　　// Product ID:
テストプログラムでは、usb_descriptors.c　内で設定されている



4）[開始]ボタン
データの送受信を開始

PCから送るデータ：65バイトの内、前半5バイトを入力
Out(1)のデータに「0」/「1」を入力すると、LED1がOff/On
Out(2)のデータに「0」/「1」を入力すると、LED2がOff/On

PCが受信するデータ：65バイトの内、前半10バイトを表示
テストボードのスイッチ1のOff/On状態が、Inp(1)のデータの「&h00」/「&h01」に対応
テストボードのスイッチ2のOff/On状態が、Inp(2)のデータの「&h00」/「&h01」に対応
Inp(2)のデータは、動作中を示すテストボード内でのカウンタ値（0x00～&hff）

【開発環境】

組み込みプログラムの開発環境を用意する

1）以下のファイルを、マイクロチップ・テクノロジー・ジャパンからダウンロード
フリーでダウンロードできるが、ファイル名は新しいバージョンになると変わる

◎MPLAB　IDE　統合環境
http://www.microchip.co.jp/download.html
例）　MPLAB_IDE_8_83.zip

◎C18コンパイラ、ユーザー登録が必要
http://www.microchip.co.jp/download.html
例）　mplabc18_v3.40_windows_lite.exe　または、mplabc18_v3.40_windows_eval.exe

◎アプリケーションライブラリ
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1486
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en547784
例）　microchip-application-libraries-v2011-12-05-windows-installer.exe

2）上記の3つのファイルを順にインストールする。

アプリケーションライブラリは、USB機能だけをインストールすればよい。

【プロジェクトの準備】

フォルダの作成、注意、ファイルパスの中に日本語（全角：2バイト文字） が無いのが良いかも。
フォルダ「hid_Custom_18」の作成
フォルダ「hid_Custom_18」の中にフォルダ「USB」の作成

ファイルのコピー。 USBに必要なファイルをコピーします。(プロジェクトとしては行儀が悪いかも)
「C:¥Microchip Solutions v2011-12-05¥USB¥Device - HID - Custom Demos¥Firmware」から、フォルダ「hid_Custom_18」にファイルをコピー
main.c
HardwareProfile.h
rm18f14k50.lkr
usb_config.h
usb_descriptors.c

「C:¥Microchip Solutions v2011-12-05¥Microchip¥USB」から、フォルダ「hid_Custom_18」にファイルをコピー
usb_device.c

「C:¥Microchip Solutions v2011-12-05¥Microchip¥USB」から、フォルダ「hid_Custom_18¥USB」にファイルをコピー
usb_hal_local.h
usb_device_local.h

「C:¥Microchip Solutions v2011-12-05¥Microchip¥USB¥HID Device Driver」から、フォルダ「hid_Custom_18」にファイルをコピー
usb_function_hid.c

「C:¥Microchip Solutions v2011-12-05¥Microchip¥Include」から、フォルダ「hid_Custom_18」にファイルをコピー
Compiler.h

「C:¥Microchip Solutions v2011-12-05¥Microchip¥Include」から、フォルダ「hid_Custom_18¥USB」にファイルをコピー
GenericTypeDefs.h

「C:¥Microchip Solutions v2011-12-05¥Microchip¥Include¥USB」から、フォルダ「hid_Custom_18¥USB」にファイルをコピー
usb.h
usb_ch9.h
usb_common.h
usb_device.h
usb_hal.h
usb_hal_pic18.h
usb_function_hid.h

「MPLAB IDE」でプロジェクトの作成
「Project」メニューから「Project Wizard」を起動
1．「Device:」で「PIC18F14K50」を選択
2．「Active Toolsuite」で「Microchip C18 Toolsuite」を選択
3．「Create New Project File」でフォルダ「hid_Custom_18」内にプロジェクトファイル「custom.mcp」名で保存
4．「Add existing files to your project」は空（Addをしない）

MPLAB IDE の「Project」ウインドウに、フォルダ「hid_Keyboard」内の「*.c *.h *.lkr」のファイルをドラッグ＆ドロップする。
「Project」ウインドウにファイル名 （拡張子で自動で振り分けられます）が表示される。

1．ファイルの修正

2．ビルド（コンパイル）のモードを「Release」にする。

3．ビルド（コンパイル）をおこなう。

4．HEXファイルをPICに書き込む

【ファイル修正】

フォルダ「cdc-basic_18」

HardwareProfile.h　　ファイルの修正、「HardwareProfile.h」を参照

main.c　　ファイルの修正、「main.c」を参照

rm18f14k50.lkr　　コメントアウトの変更（今回のテスト回路とプログラムでは変更しなくてもよい）
修正後　//Bootloader
・　CODEPAGE　NAME=vectors　START=0x0　END=0x29　PROTECTED
→　//CODEPAGE　NAME=bootloader　START=0x2A　END=0xFFF　PROTECTED
・　CODEPAGE　NAME=page　START=0x1000　END=0x3FFF

usb_config.h　　コメントアウトの変更
制御方式を、ポーリング形式の場合
#define USB_POLLING
//#define USB_INTERRUPT
または
制御方式を、割込みイベント形式の場合
//#define USB_POLLING
#define USB_INTERRUPT

usb_device.c　　ファイルパスの変更
修正前　#include "../USB/usb_device_local.h"
修正後　#include "./USB/usb_device_local.h"

usb_function_cdc.c　　変更なし
usb_descriptors.c　　変更なし
Compiler.h　　変更なし

「cdc-basic_18¥USB」

usb_hal_local.h　　変更なし
usb_device_local.h　　変更なし
GenericTypeDefs.h　　変更なし
usb.h　　変更なし
usb_ch9.h　　変更なし
usb_common.h　　変更なし
usb_device.h　　変更なし
usb_hal.h　　変更なし
usb_hal_pic18.h　　変更なし
usb_function_cdc.h　　変更なし

【HardwareProfile.h】

ファイル全体を以下の内容に置換える

```
/********************************************************************
 FileName:      HardwareProfile.h
********************************************************************/

#ifndef HARDWARE_PROFILE_H
#define HARDWARE_PROFILE_H

    /*******************************************************************/
    /******** USB stack hardware selection options *********************/
    /*******************************************************************/
#if defined(__PIC24FJ64GB002__)    // PIC24FJ64GB002
    //#define USE_SELF_POWER_SENSE_IO
    #define tris_self_power     TRISAbits.TRISA2    // Input
    #define self_power          1

    //#define USE_USB_BUS_SENSE_IO
    #define tris_usb_bus_sense  U1OTGSTATbits.SESVD  // Input
    #define USB_BUS_SENSE       U1OTGSTATbits.SESVD
#endif

#if defined(__18F14K50)
    //#define USE_SELF_POWER_SENSE_IO
    #define tris_self_power     TRISCbits.TRISC2    // Input
    #define self_power          1

    //#define USE_USB_BUS_SENSE_IO
    #define tris_usb_bus_sense  TRISCbits.TRISC2    // Input
    #define USB_BUS_SENSE       1
#endif

    /*******************************************************************/
    /******** Application specific definitions *************************/
    /*******************************************************************/
#if defined(__PIC24FJ64GB002__)  // PIC24FJ64GB002

    #define PIC24F_STARTER_KIT
    #define CLOCK_FREQ 32000000

    /** SWITCH **/
    #define mInitSwitch1()      TRISBbits.TRISB0=1;
    #define mInitSwitch2()      TRISBbits.TRISB1=1;
    #define mInitAllSwitches()  mInitSwitch1();mInitSwitch2();
    #define sw1                 PORTBbits.RB0
    #define sw2                 PORTBbits.RB1

    #define led01               LATAbits.LATA0
    #define led02               LATAbits.LATA1
#endif

#if defined(__18F14K50)
    #define DEMO_BOARD PIC18F_STARTER_KIT_1
    #define CLOCK_FREQ 48000000
    #define GetSystemClock() CLOCK_FREQ

    /** SWITCH **/
    #define mInitSwitch1()      TRISCbits.TRISC4=1;
    #define mInitSwitch2()      TRISCbits.TRISC5=1;
    #define mInitAllSwitches()  mInitSwitch1();mInitSwitch2();
    #define sw1                 PORTCbits.RC4
    #define sw2                 PORTCbits.RC5

    #define led01               LATBbits.LATB4
    #define led02               LATBbits.LATB5
#endif


    /** I/O pin definitions **/
    #define INPUT_PIN 1
    #define OUTPUT_PIN 0


#endif  //HARDWARE_PROFILE_H
```

【main.c】

プログラムの修正
・置換え箇所
最初の行から、コメント部分「/******* USB Callback Functions **********/」の前までを、
緑色の行の内容に置換える。

・修正箇所
コメント部分「/******* USB Callback Functions **********/」以降の修正なし。

```
#ifndef MAIN_C
#define MAIN_C

/** INCLUDES *******************************************************/
#if defined(__PIC24F__)
#include <p24fj64gb002.h>
#endif

#if defined(__18F14K50)
#include <p18f14k50.h>
#endif

#include "./USB/usb.h"
#include "HardwareProfile.h"
#include "./USB/usb_function_hid.h"

/** CONFIGURATION **************************************************/
#if defined(__PIC24FJ64GB002__)
_CONFIG1(WINDIS_OFF & FWDTEN_OFF & ICS_PGx1 & GCP_OFF & JTAGEN_OFF)
_CONFIG2(IESO_ON & PLLDIV_DIV2 & PLL96MHZ_ON & FNOSC_FRCPLL & FCKSM_CSDCMD & OSCIOFNC_ON &
         IOL1WAY_OFF & I2C1SEL_PRI & POSCMOD_NONE)
_CONFIG3(WPFP_WPFP0 & SOSCSEL_IO & WUTSEL_LEG & WPDIS_WPDIS & WPCFG_WPCFGDIS & WPEND_WPENDMEM)
_CONFIG4(DSWDTPS_DSWDTPS3 & DSWDTOSC_LPRC & RTCOSC_SOSC & DSBOREN_OFF & DSWDTEN_OFF)
#endif

#if defined(__18F14K50)
#pragma config CPUDIV = NOCLKDIV ,USBDIV = OFF, FOSC = HS
#pragma config PLLEN = OFF, PCLKEN = ON, HFOFST = OFF, DEBUG = OFF
#pragma config PWRTEN = ON, BOREN = OFF, BORV = 30, MCLRE = ON
#pragma config FCMEN = OFF, IESO = OFF, WDTEN = OFF, WDTPS = 1, LVP = OFF
#pragma config XINST = OFF, STVREN = ON, BBSIZ = OFF
#pragma config CP0 = OFF, CP1 = OFF, CPB = OFF, CPD = OFF
#pragma config WRT0 = OFF,  WRT1 = OFF, WRTC = OFF, WRTB = OFF, WRTD = OFF
#pragma config EBTR0 = OFF, EBTR1 = OFF, EBTRB = OFF
#endif
/** VARIABLES ******************************************************/
#pragma udata
BYTE old_sw1,old_sw2;

#if defined(__18F14K50)
    #pragma udata usbram2
#else
    #pragma udata
#endif

unsigned char ReceivedDataBuffer[64];
unsigned char ToSendDataBuffer[64];

#pragma udata
USB_HANDLE USBOutHandle = 0;
USB_HANDLE USBInHandle = 0;

/** PRIVATE PROTOTYPES *********************************************/
void BlinkUSBStatus(void);
BOOL Switch1IsPressed(void);
BOOL Switch2IsPressed(void);
static void InitializeSystem(void);
void ProcessIO(void);
void UserInit(void);

void USBCBSendResume(void);

/** VECTOR REMAPPING ***********************************************/
#pragma code
#if defined(USB_INTERRUPT)
    #if defined(__18CXX)
        void YourHighPriorityISRCode();
        void YourLowPriorityISRCode();

        #define REMAPPED_RESET_VECTOR_ADDRESS            0x00
        #define REMAPPED_HIGH_INTERRUPT_VECTOR_ADDRESS    0x08
        #define REMAPPED_LOW_INTERRUPT_VECTOR_ADDRESS    0x18

        #pragma code REMAPPED_HIGH_INTERRUPT_VECTOR = REMAPPED_HIGH_INTERRUPT_VECTOR_ADDRESS
        void Remapped_High_ISR (void)
        {
            _asm goto YourHighPriorityISRCode _endasm
```

```
        }
        #pragma code REMAPPED_LOW_INTERRUPT_VECTOR = REMAPPED_LOW_INTERRUPT_VECTOR_ADDRESS
        void Remapped_Low_ISR (void)
        {
            _asm goto YourLowPriorityISRCode _endasm
        }

        #pragma code
        #pragma interrupt YourHighPriorityISRCode
        void YourHighPriorityISRCode()
        {
            USBDeviceTasks();
        }
        #pragma interruptlow YourLowPriorityISRCode
        void YourLowPriorityISRCode()
        {
        }

    #endif
#endif


#if 0
void __attribute__((interrupt, auto_psv)) _T1Interrupt(void)
{
}
#endif
```

```
/** DECLARATIONS ***************************************************/
#pragma code

/** DECLARATIONS ***************************************************/
#pragma code

/*********************************************************************
 * Function:        void main(void)
 *********************************************************************/
void main(void)
{
    InitializeSystem();

    USBDeviceInit(); //usb_device.c.
                     // Initializes USB module SFRs and firmware variables to known states.

    #if defined(USB_INTERRUPT)
        USBDeviceAttach();
    #endif

    while(1)
    {

    #if defined(USB_POLLING)
        // Check bus status and service USB interrupts.
        USBDeviceTasks();
    #endif

        // Application-specific tasks.
        // Application related code may be added here, or in the ProcessIO() function.
        ProcessIO();
    }
}
```

```
/*********************************************************************
 * Function:        static void InitializeSystem(void)
 *********************************************************************/
static void InitializeSystem(void)
{
#if defined(__PIC24FJ64GB002__)
    unsigned int pll_startup_counter = 600;

    CLKDIV = 0x0000;                    // CPU:32MHz
    CLKDIVbits.PLLEN = 1;               // 96MHz PLL On,
    while(pll_startup_counter--);

    RCONbits.SWDTEN = 0;                //ウオッチドック OFF
    AD1PCFG = 0xFFFF;                   //AD OFF


    /* 入出力ポート設定  */
    TRISA = 0x0000;                     //ポートAを全て出力
    TRISB = 0x0003;                     //ボートBの0&1ビット目を入力、他は入力
#endif

#if defined(__18F14K50)
        /* 入出力ポート設定  */
    TRISA = 0x00;
    TRISB = 0x00;
    TRISC = 0xf0;
```

```
    ADCON0bits.ADON = 0;    //AD OFF
#endif

    //Initialize all of the push buttons
    mInitAllSwitches();
    old_sw1 = sw1;
    old_sw2 = sw2;

    //initialize the variable holding the handle for the last transmission
    USBOutHandle = 0;
    USBInHandle = 0;

    /*USBの状態検出の指定*/
    #if defined(USE_USB_BUS_SENSE_IO)
        tris_usb_bus_sense = INPUT_PIN; // See HardwareProfile.h
    #endif

    #if defined(USE_SELF_POWER_SENSE_IO)
        tris_self_power = INPUT_PIN;    // See HardwareProfile.h
    #endif
}


/********************************************************************
 * Function:        void ProcessIO(void)
 ********************************************************************/
void ProcessIO(void)
{
// User Application USB tasks
if((USBDeviceState < CONFIGURED_STATE)||(USBSuspendControl==1)) return;



if( HIDRxHandleBusy(USBOutHandle) == FALSE )      //受信中？ TRUE:ビジー状態  FALSE:受信終了
{
    USBOutHandle = HIDRxPacket(HID_EP,(BYTE*)&ReceivedDataBuffer,64);      //パケットの受信
    led01 = 1;          //LED1 Off
    if( (ReceivedDataBuffer[0] & 0x01) == 0x01 )
    {
        led01 = 0;      //LED1 On
    }

    led02 = 1;          //LED2 Off
       if( (ReceivedDataBuffer[1] & 0x01) == 0x01 )
    {
        led02 = 0;      //LED2 On
    }

}


if( HIDTxHandleBusy(USBInHandle) == FALSE )       //送信中？ TRUE:ビジー状態  FALSE:送信終了
    {
    ToSendDataBuffer[0] = 0;
    if (sw1 == 0 )          //SW ON?  0:On 1:Off
    {
        ToSendDataBuffer[0] =  0x01;
    }

    ToSendDataBuffer[1] = 0;
    if (sw2 == 0 )      //SW ON?  0:On 1:Off
        {
            ToSendDataBuffer[1] = 0x01;
        }

        ToSendDataBuffer[2] = ToSendDataBuffer[2] + 1;

        USBInHandle = HIDTxPacket(HID_EP,(BYTE*)&ToSendDataBuffer[0],64);
    }

}


/********************************************************************
 * Function:        BOOL Switch1IsPressed(void)
 ********************************************************************/
BOOL Switch1IsPressed(void)
{
    if(sw1 != old_sw1)
    {
        old_sw1 = sw1;                  // Save new value
        if(sw1 == 0)                    // If pressed
            return TRUE;                // Was pressed
    }
    return FALSE;                       // Was not pressed
}

/********************************************************************
 * Function:        BOOL Switch2IsPressed(void)
```

```
 ***************************************************************************/
BOOL Switch2IsPressed(void)
{
    if(sw2 != old_sw2)
    {
        old_sw2 = sw2;                  // Save new value
        if(sw2 == 0)                    // If pressed
            return TRUE;                // Was pressed
    }
    return FALSE;                       // Was not pressed
}


// ****************************************************************************************
// ************** USB Callback Functions ***************************************************
// ****************************************************************************************
```

以下は変更無し、そのまま残す

【通信部分の関数説明】

・HIDのUSB通信バッファの大きさは、受信：64バイト、送信：64バイト

受信バッファ
```
unsigned char ReceivedDataBuffer[64];
```
送信バッファ
```
unsigned char ToSendDataBuffer[64];
```

・パケットの受信は以下の関数でおこなう

```
USBOutHandle = HIDRxPacket(HID_EP,(BYTE*)&ReceivedDataBuffer,64);
```

引数1：HID_EP：使用するエンドポイントの番号
引数2：(BYTE*)&ReceivedDataBuffer：受信文字列のバッファポインタ
引数3：64：受信文字数

戻り値：USBOutHandle：受信完了チェックに用いるハンドル番号

```
BOOL HIDRxHandleBusy(USBOutHandle)
```

引数1：USBOutHandle:先のHIDRxPacket()で返ってきたハンドル番号

戻り値：BOOL:TRUE=ビジー状態、FALSE=受信終了

・パケットの送信は以下の関数でおこなう

```
USBInHandle = HIDTxPacket(HID_EP,(BYTE*)&ToSendDataBuffer[0],64);
```

引数1：HID_EP：使用するエンドポイントの番号
引数2：(BYTE*)&ToSendDataBuffer[0]：送信文字列のバッファポインタ
引数3：64：送信文字数

戻り値：USBOutHandle：受信完了チェックに用いるハンドル番号

```
BOOL HIDTxHandleBusy(USBInHandle)
```

引数1：USBInHandle:先のHIDTxPacket()で返ってきたハンドル番号

戻り値：BOOL:TRUE=ビジー状態、FALSE=送信終了

【備考、CONFIG設定】

システムクロックを変更する場合は、データシートを参照して　CONFIG　を修正する。
PIC18F13K50/14K50 Data Sheet　(DS41350C-page 20)

2.11 USB Operation
　TABLE 2-4: LOW SPEED USB CLOCK SETTINGS　　の表を参照すること
　TABLE 2-5: FULL-SPEED USB CLOCK SETTINGS　　の表を参照すること

「C:¥Program Files¥Microchip¥mplabc18¥v3.40¥mpasm¥P18F14K50.INC」より抜粋

```
;==========================================================================
;
;   IMPORTANT: For the PIC18 devices, the __CONFIG directive has been
;              superseded by the CONFIG directive.  The following settings
;              are available for this device.
;
;   CPU System Clock Selection bits:
;     CPUDIV = NOCLKDIV    No CPU System Clock divide
;     CPUDIV = CLKDIV2     CPU System Clock divided by 2
;     CPUDIV = CLKDIV3     CPU System Clock divided by 3
;     CPUDIV = CLKDIV4     CPU System Clock divided by 4
;
;   USB Clock Selection bit:
;     USBDIV = OFF         USB clock comes directly from the OSC1/OSC2 oscillator block; no divide
;     USBDIV = ON          USB clock comes from the OSC1/OSC2 divided by 2
;
;   Oscillator Selection bits:
;     FOSC = LP            LP oscillator
;     FOSC = XT            XT oscillator
;     FOSC = HS            HS oscillator
;     FOSC = ERCCLKOUT     External RC oscillator, CLKOUT function on OSC2
;     FOSC = ECCLKOUTH     EC, CLKOUT function on OSC2 (high)
;     FOSC = ECH           EC (high)
;     FOSC = ERC           External RC oscillator
;     FOSC = IRC           Internal RC oscillator
;     FOSC = IRCCLKOUT     Internal RC oscillator, CLKOUT function on OSC2
;     FOSC = ECCLKOUTM     EC, CLKOUT function on OSC2 (medium)
;     FOSC = ECM           EC (medium)
;     FOSC = ECCLKOUTL     EC, CLKOUT function on OSC2 (low)
;     FOSC = ECL           EC (low)
;
;   4 X PLL Enable bit:
;     PLLEN = OFF          PLL is under software control
;     PLLEN = ON           Oscillator multiplied by 4
;
;   Primary Clock Enable bit:
;     PCLKEN = OFF         Primary clock is under software control
;     PCLKEN = ON          Primary clock enabled
;
;   Fail-Safe Clock Monitor Enable:
;     FCMEN = OFF          Fail-Safe Clock Monitor disabled
;     FCMEN = ON           Fail-Safe Clock Monitor enabled
;
;   Internal/External Oscillator Switchover bit:
;     IESO = OFF           Oscillator Switchover mode disabled
;     IESO = ON            Oscillator Switchover mode enabled
;
;   Power-up Timer Enable bit:
;     PWRTEN = ON          PWRT enabled
;     PWRTEN = OFF         PWRT disabled
;
;   Brown-out Reset Enable bits:
;     BOREN = OFF          Brown-out Reset disabled in hardware and software
;     BOREN = ON           Brown-out Reset enabled and controlled by software (SBOREN is enabled)
;     BOREN = NOSLP        Brown-out Reset enabled in hardware only and disabled in Sleep mode (SBOREN is disabled)
;     BOREN = SBORDIS      Brown-out Reset enabled in hardware only (SBOREN is disabled)
;
;   Brown-out Reset Voltage bits:
;     BORV = 30            VBOR set to 3.0 V nominal
;     BORV = 27            VBOR set to 2.7 V nominal
;     BORV = 22            VBOR set to 2.2 V nominal
;     BORV = 19            VBOR set to 1.9 V nominal
;
;   Watchdog Timer Enable bit:
;     WDTEN = OFF          WDT is controlled by SWDTEN bit of the WDTCON register
;     WDTEN = ON           WDT is always enabled. SWDTEN bit has no effect.
;
;   Watchdog Timer Postscale Select bits:
;     WDTPS = 1            1:1
;     WDTPS = 2            1:2
;     WDTPS = 4            1:4
;     WDTPS = 8            1:8
;     WDTPS = 16           1:16
;     WDTPS = 32           1:32
```

```
;     WDTPS = 64          1:64
;     WDTPS = 128         1:128
;     WDTPS = 256         1:256
;     WDTPS = 512         1:512
;     WDTPS = 1024        1:1024
;     WDTPS = 2048        1:2048
;     WDTPS = 4096        1:4096
;     WDTPS = 8192        1:8192
;     WDTPS = 16384       1:16384
;     WDTPS = 32768       1:32768
;
;   HFINTOSC Fast Start-up bit:
;     HFOFST = OFF        The system clock is held off until the HFINTOSC is stable.
;     HFOFST = ON         HFINTOSC starts clocking the CPU without waiting for the oscillator to stablize.
;
;   MCLR Pin Enable bit:
;     MCLRE = OFF         RA3 input pin enabled; MCLR disabled
;     MCLRE = ON          MCLR pin enabled; RA3 input pin disabled
;
;   Stack Full/Underflow Reset Enable bit:
;     STVREN = OFF        Stack full/underflow will not cause Reset
;     STVREN = ON         Stack full/underflow will cause Reset
;
;   Single-Supply ICSP Enable bit:
;     LVP = OFF           Single-Supply ICSP disabled
;     LVP = ON            Single-Supply ICSP enabled
;
;   Boot Block Size Select bit:
;     BBSIZ = OFF         1kW boot block size
;     BBSIZ = ON          2kW boot block size
;
;   Extended Instruction Set Enable bit:
;     XINST = OFF         Instruction set extension and Indexed Addressing mode disabled (Legacy mode)
;     XINST = ON          Instruction set extension and Indexed Addressing mode enabled
;
;   Background Debugger Enable bit:
;     DEBUG = ON          Background debugger enabled, RA0 and RA1 are dedicated to In-Circuit Debug
;     DEBUG = OFF         Background debugger disabled, RA0 and RA1 configured as general purpose I/O pins
;
;   Code Protection bit:
;     CP0 = ON            Block 0 code-protected
;     CP0 = OFF           Block 0 not code-protected
;
;   Code Protection bit:
;     CP1 = ON            Block 1 code-protected
;     CP1 = OFF           Block 1 not code-protected
;
;   Boot Block Code Protection bit:
;     CPB = ON            Boot block code-protected
;     CPB = OFF           Boot block not code-protected
;
;   Data EEPROM Code Protection bit:
;     CPD = ON            Data EEPROM code-protected
;     CPD = OFF           Data EEPROM not code-protected
;
;   Table Write Protection bit:
;     WRT0 = ON           Block 0 write-protected
;     WRT0 = OFF          Block 0 not write-protected
;
;   Table Write Protection bit:
;     WRT1 = ON           Block 1 write-protected
;     WRT1 = OFF          Block 1 not write-protected
;
;   Configuration Register Write Protection bit:
;     WRTC = ON           Configuration registers write-protected
;     WRTC = OFF          Configuration registers not write-protected
;
;   Boot Block Write Protection bit:
;     WRTB = ON           Boot block write-protected
;     WRTB = OFF          Boot block not write-protected
;
;   Data EEPROM Write Protection bit:
;     WRTD = ON           Data EEPROM write-protected
;     WRTD = OFF          Data EEPROM not write-protected
;
;   Table Read Protection bit:
;     EBTR0 = ON          Block 0 protected from table reads executed in other blocks
;     EBTR0 = OFF         Block 0 not protected from table reads executed in other blocks
;
;   Table Read Protection bit:
;     EBTR1 = ON          Block 1 protected from table reads executed in other blocks
;     EBTR1 = OFF         Block 1 not protected from table reads executed in other blocks
;
;   Boot Block Table Read Protection bit:
;     EBTRB = ON          Boot block protected from table reads executed in other blocks
;     EBTRB = OFF         Boot block not protected from table reads executed in other blocks
;
;==========================================================================
```