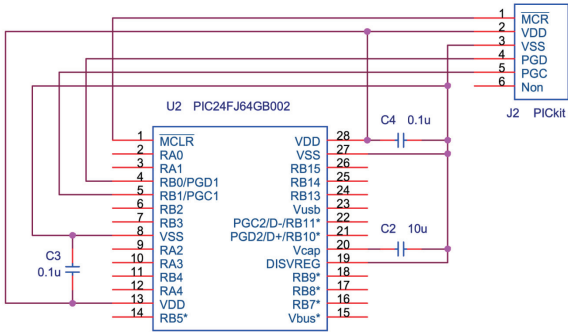


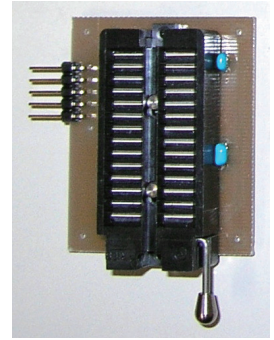
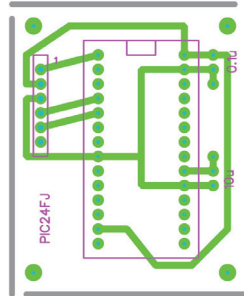
【回路図】

PIC24F64Gx002 書き込みアダプタ

回路図

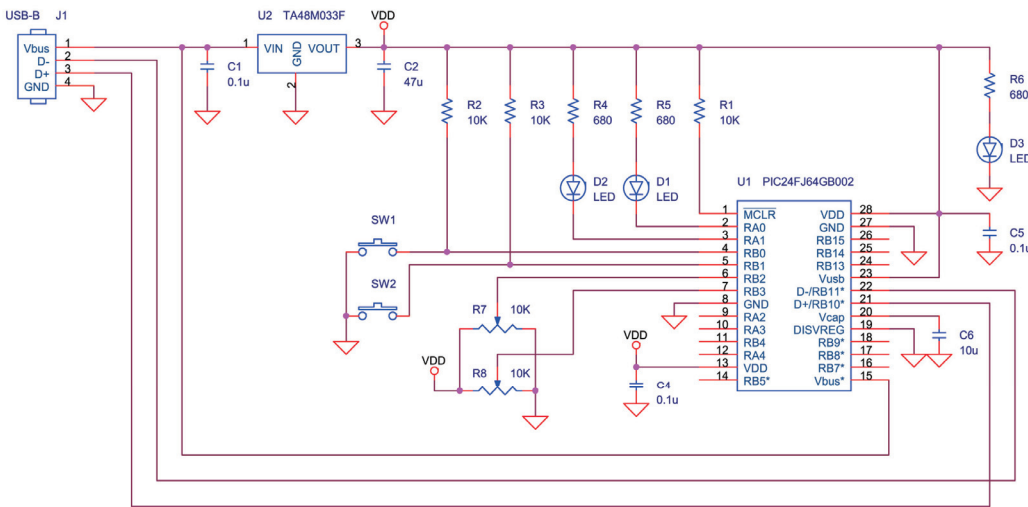


部品面



PIC24F64GB002 USBテスト回路

回路図

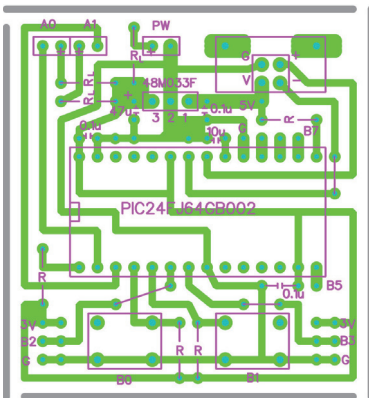


記号	名称	個数
USB-B	USBコネクタBタイプ	1
C4,C5	セラミックコンデンサ 0.1μF	1
C6	セラミックコンデンサ 10μF	1
R1~R3	抵抗 10KΩ	2
R4~R6	抵抗 680Ω~1KΩ	3
R7,R8	可変抵抗 5KΩ~10KΩ	2
D1~D3	LED	3
SW1~SW2	押しボタンスイッチ 押したときだけON	2
U1	ゼロブレッシングャー (ZIF)	1
	ICソケット	1
	PIC24F64GB002	1
U2	5V → 3.3V レギュレータ	1
C1	セラミックコンデンサ 0.1μF	1
C2	電界コンデンサ 47μF	1

C1,C2は使用するレギュレータに合わせる

可変抵抗は、アナログ入力へのテスト用、今後のテストプログラムで使用できる

部品面

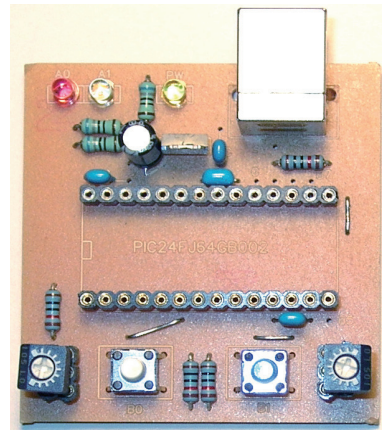


PICのICソケットには、ZIFソケットを使用

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)

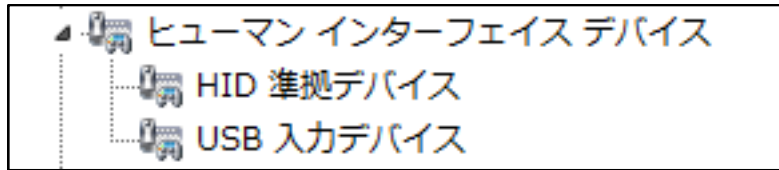
注意: R0とRB1などは略して、R0とR1と記している。



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]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

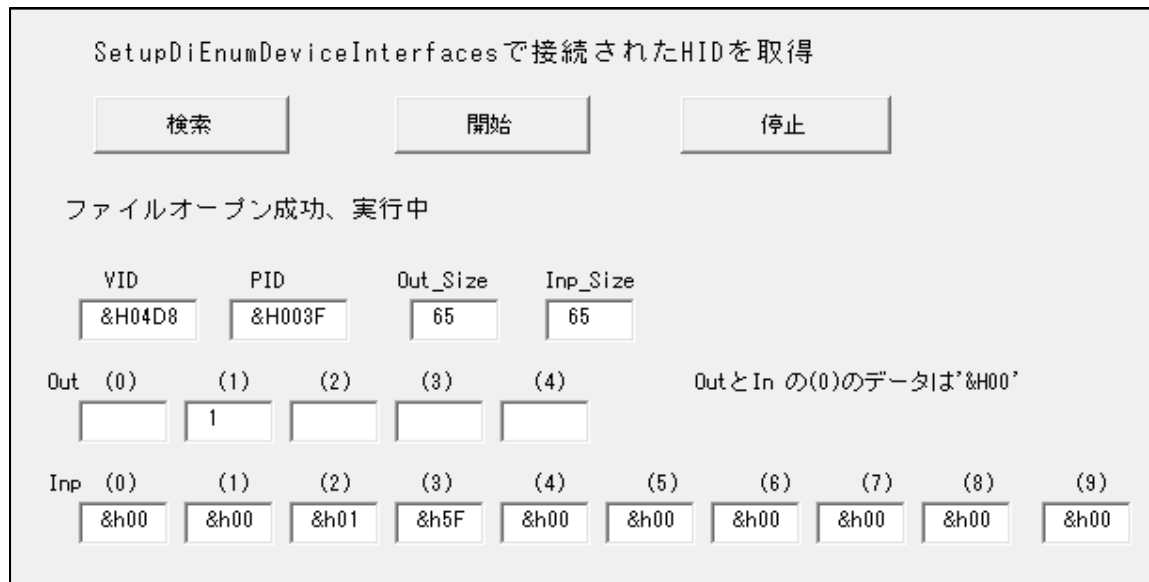
A	B	C	D
	&h04D8	&h0033	
	&h04D8	&h003F	

- 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

◎C30コンパイラ、ユーザー登録が必要

<http://www.microchip.co.jp/download.html>

例) mplabc30_v3_30c_windows.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_24」を作成

フォルダ「hid_Custom_24」の中にフォルダ「USB」を作成

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_24」にファイルをコピー

main.c

HardwareProfile.h

usb_config.h

usb_descriptors.c

「C:\Microchip Solutions v2011-12-05\Microchip\USB」から、フォルダ「hid_Custom_24」にファイルをコピー

usb_device.c

usb_hal_pic24.c

「C:\Microchip Solutions v2011-12-05\Microchip\USB」から、フォルダ「hid_Custom_24\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_24」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_24」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_24\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_24\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_pic24.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「PIC24FJ64GB002」を選択
2. 「Active Toolsuite」で「Microchip C30 Toolsuite」を選択
3. 「Create New Project File」でフォルダ「hid_Custom_24」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_Joystick」内の「*.c *.h」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「hid_Custom_24」

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

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、割り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_hid.c
ファイルパスの変更
修正前 #include "GenericTypeDefs.h"
修正後 #include "../USB/GenericTypeDefs.h"

usb_descriptors.c
Compiler.h
usb_hal_pic24.c
変更なし
変更なし
変更なし

「hid_Custom_24」USB

usb_hal_pic24.h
ファイルパスの変更
修正前 #include "Compiler.h"
修正後 #include "../Compiler.h"

修正前 #include "usb_config.h"
修正後 #include "../usb_config.h"

usb_hal_local.h
ファイルパスの変更
修正前 #include "usb_config.h"
修正後 #include "../usb_config.h"

usb.h
ファイルパスの変更
修正前 #include "Compiler.h"
・ #include "usb_config.h" // Must be defined by the application
・ #include "USB/usb_common.h" // Common USB library definitions
・ #include "USB/usb_ch9.h" // USB device framework definitions
・ #if defined(USB_SUPPORT_DEVICE)
・ #include "USB/usb_device.h" // USB Device abstraction layer interface
・ #endif
・ #include "USB/usb_hal.h" // Hardware Abstraction Layer interface
修正後 #include "../Compiler.h"
・ #include "../usb_config.h" // Must be defined by the application
・ #include "usb_common.h" // Common USB library definitions
・ #include "usb_ch9.h" // USB device framework definitions
・ #if defined(USB_SUPPORT_DEVICE)
・ #include "usb_device.h" // USB Device abstraction layer interface
・ #endif
・ #include "usb_hal.h" // Hardware Abstraction Layer interface

修正前 #include "USB/usb_hal_pic24.h"
修正後 #include "usb_hal_pic24.h"

usb_device_local.h
GenericTypeDefs.h
usb_ch9.h
usb_common.h
usb_device.h
usb_hal.h
usb_function_hid.h
変更なし
変更なし
変更なし
変更なし
変更なし
変更なし
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    U10TGSTATbits.SESVD // Input
    #define USB_BUS_SENSE        U10TGSTATbits.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 3200000

    /** 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 4800000
    #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 & SOSSEL_IO & WUTSEL_LEG & WPDIS_WPDIS & WPCFG_WPCFGDIS & WPEND_WPENDMEM)
_CONFIG4(DSWDTPS_DSWDTPS3 & DSWDTOSC_LPRC & RTCOSC_SOSC & DSBORN_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, WDTEEN = OFF, WDTPS = 1, LVP = OFF
#pragma config XINST = OFF, STVREN = ON, BBSIZ = OFF
#pragma config CPO = 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
}

#endif
#endif

```

```

    }
    #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.PLEN = 1;       // 96MHz PLL On,
        while(pll_startup_counter--);

        RCONbits.SWDTEN = 0;       //ウォッチドック 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;           //ポート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;
    #endif
}

```



```

    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_I0)
    tris_usb_bus_sense = INPUT_PIN; // See HardwareProfile.h
#endif

#if defined(USE_SELF_POWER_SENSE_I0)
    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】

「C:\Program Files\Microchip\MPLAB C30\support\PIC24F\inc\p24FJ64GB002.inc」より抜粋

```
----- CONFIG4 (0xabf8) -----
;
; The following settings are available for CONFIG4:
;
; DSWDT Postscale Select:
; DSWDTPS_DSWDTPS0    1:2 (2.1 ms)
; DSWDTPS_DSWDTPS1    1:8 (8.3 ms)
; DSWDTPS_DSWDTPS2    1:32 (33 ms)
; DSWDTPS_DSWDTPS3    1:128 (132 ms)
; DSWDTPS_DSWDTPS4    1:512 (528 ms)
; DSWDTPS_DSWDTPS5    1:2,048 (2.1 seconds)
; DSWDTPS_DSWDTPS6    1:8,192 (8.5 seconds)
; DSWDTPS_DSWDTPS7    1:32,768 (34 seconds)
; DSWDTPS_DSWDTPS8    1:131,072 (135 seconds)
; DSWDTPS_DSWDTPS9    1:524,288 (9 minutes)
; DSWDTPS_DSWDTPSA    1:2,097,152 (36 minutes)
; DSWDTPS_DSWDTPSB    1:8,388,608 (2.4 hours)
; DSWDTPS_DSWDTPSC    1:33,554,432 (9.6 hours)
; DSWDTPS_DSWDTPSD    1:134,217,728 (38.5 hours)
; DSWDTPS_DSWDTPSE    1:536,870,912 (6.4 days)
; DSWDTPS_DSWDTPSF    1:2,147,483,648 (25.7 days)
;
; Deep Sleep Watchdog Timer Oscillator Select:
; DSWDTOSC_SOSC       DSWDT uses Secondary Oscillator (SOSC)
; DSWDTOSC_LPRC       DSWDT uses Low Power RC Oscillator (LPRC)
;
; RTCC Reference Oscillator Select:
; RTCOSC_LPRC         RTCC uses Low Power RC Oscillator (LPRC)
; RTCOSC_SOSC         RTCC uses Secondary Oscillator (SOSC)
;
; Deep Sleep BOR Enable bit:
; DSBOREN_OFF         BOR disabled in Deep Sleep
; DSBOREN_ON          BOR enabled in Deep Sleep
;
; Deep Sleep Watchdog Timer:
; DSWDTEN_OFF         DSWDT disabled
; DSWDTEN_ON          DSWDT enabled
;
----- CONFIG3 (0xabfa) -----
;
; The following settings are available for CONFIG3:
;
; Write Protection Flash Page Segment Boundary:
; WFPF_WFPF0          Page 0 (0x0)
; WFPF_WFPF1          Page 1 (0x400)
; WFPF_WFPF2          Page 2 (0x800)
; WFPF_WFPF3          Page 3 (0xC00)
; WFPF_WFPF4          Page 4 (0x1000)
; WFPF_WFPF5          Page 5 (0x1400)
; WFPF_WFPF6          Page 6 (0x1800)
; WFPF_WFPF7          Page 7 (0x1C00)
; WFPF_WFPF8          Page 8 (0x2000)
; WFPF_WFPF9          Page 9 (0x2400)
; WFPF_WFPF10         Page 10 (0x2800)
; WFPF_WFPF11         Page 11 (0x2C00)
; WFPF_WFPF12         Page 12 (0x3000)
; WFPF_WFPF13         Page 13 (0x3400)
; WFPF_WFPF14         Page 14 (0x3800)
```

```

; WFPF_WFPF15      Page 15 (0x3C00)
; WFPF_WFPF16      Page 16 (0x4000)
; WFPF_WFPF17      Page 17 (0x4400)
; WFPF_WFPF18      Page 18 (0x4800)
; WFPF_WFPF19      Page 19 (0x4C00)
; WFPF_WFPF20      Page 20 (0x5000)
; WFPF_WFPF21      Page 21 (0x5400)
; WFPF_WFPF22      Page 22 (0x5800)
; WFPF_WFPF23      Page 23 (0x5C00)
; WFPF_WFPF24      Page 24 (0x6000)
; WFPF_WFPF25      Page 25 (0x6400)
; WFPF_WFPF26      Page 26 (0x6800)
; WFPF_WFPF27      Page 27 (0x6C00)
; WFPF_WFPF28      Page 28 (0x7000)
; WFPF_WFPF29      Page 29 (0x7400)
; WFPF_WFPF30      Page 30 (0x7800)
; WFPF_WFPF31      Page 31 (0x7C00)
; WFPF_WFPF32      Page 32 (0x8000)
; WFPF_WFPF33      Page 33 (0x8400)
; WFPF_WFPF34      Page 34 (0x8800)
; WFPF_WFPF35      Page 35 (0x8C00)
; WFPF_WFPF36      Page 36 (0x9000)
; WFPF_WFPF37      Page 37 (0x9400)
; WFPF_WFPF38      Page 38 (0x9800)
; WFPF_WFPF39      Page 39 (0x9C00)
; WFPF_WFPF40      Page 40 (0xA000)
; WFPF_WFPF41      Page 41 (0xA400)
; WFPF_WFPF42      Page 42 (0xA800)
; WFPF_WFPF63      Highest Page (same as page 42)
;
; Secondary Oscillator Pin Mode Select:
; SOSCSEL_IO       SOSC pins have digital I/O functions (RA4, RB4)
; SOSCSEL_LPSOSC   SOSC pins in Low-Power (low drive-strength) Oscillator Mode
; SOSCSEL_SOSC     SOSC pins in Default (high drive-strength) Oscillator Mode
;
; Voltage Regulator Wake-up Time Select:
; WUTSEL_FST       Fast regulator start-up time used
; WUTSEL_LEG       Default regulator start-up time used
;
; Segment Write Protection Disable:
; WPDIS_WPEN       Segmented code protection enabled
; WPDIS_WPDIS      Segmented code protection disabled
;
; Write Protect Configuration Page Select:
; WPCFG_WPCFGGEN   Last page and Flash Configuration words are code-protected
; WPCFG_WPCFGDIS   Last page and Flash Configuration words are unprotected
;
; Segment Write Protection End Page Select:
; WPEND_WPSTARTMEM Write Protect from page 0 to WFPF
; WPEND_WPENDMEM   Write Protect from WFPF to the last page of memory
;
;----- CONFIG2 (0xabfc) -----
;
; The following settings are available for CONFIG2:
;
; Primary Oscillator Select:
; POSCMOD_EC       EC Oscillator mode selected
; POSCMOD_XT       XT Oscillator mode selected
; POSCMOD_HS       HS Oscillator mode selected
; POSCMOD_NONE     Primary Oscillator disabled
;
; I2C1 Pin Select bit:

```

```

; I2C1SEL_SEC      Use alternate SCL1/SDA1 pins for I2C1
; I2C1SEL_PRI      Use default SCL1/SDA1 pins for I2C1
;
; IOLOCK One-Way Set Enable:
; IOL1WAY_OFF      The IOLOCK bit can be set and cleared using the unlock sequence
; IOL1WAY_ON        Once set, the IOLOCK bit cannot be cleared
;
; OSCO Pin Configuration:
; OSCIOFNC_ON       OSCO pin functions as port I/O (RA3)
; OSCIOFNC_OFF      OSCO pin functions as clock output (CLKO)
;
; Clock Switching and Fail-Safe Clock Monitor:
; FCKSM_CSECME      Sw Enabled, Mon Enabled
; FCKSM_CSECMD      Sw Enabled, Mon Disabled
; FCKSM_CSDCMD      Sw Disabled, Mon Disabled
;
; Initial Oscillator Select:
; FNOSC_FRC          Fast RC Oscillator (FRC)
; FNOSC_FRCPLL       Fast RC Oscillator with Postscaler and PLL module (FRCPLL)
; FNOSC_PRI          Primary Oscillator (XT, HS, EC)
; FNOSC_PRIPLL       Primary Oscillator with PLL module (XTPLL, HSPLL, ECPLL)
; FNOSC_SOSC         Secondary Oscillator (SOSC)
; FNOSC_LPRC         Low-Power RC Oscillator (LPRC)
; FNOSC_FRCDIV       Fast RC Oscillator with Postscaler (FRCDIV)
;
; 96MHz PLL Startup Select:
; PLL96MHZ_OFF      96 MHz PLL Startup is enabled by user in software( controlled with the PLEN bit)
; PLL96MHZ_ON       96 MHz PLL Startup is enabled automatically on start-up
;
; USB 96 MHz PLL Prescaler Select:
; PLLDIV_NODIV      Oscillator input used directly (4 MHz input)
; PLLDIV_DIV2        Oscillator input divided by 2 (8 MHz input)
; PLLDIV_DIV3        Oscillator input divided by 3 (12 MHz input)
; PLLDIV_DIV4        Oscillator input divided by 4 (16 MHz input)
; PLLDIV_DIV5        Oscillator input divided by 5 (20 MHz input)
; PLLDIV_DIV6        Oscillator input divided by 6 (24 MHz input)
; PLLDIV_DIV8        Oscillator input divided by 8 (32 MHz input)
; PLLDIV_DIV12       Oscillator input divided by 12 (48 MHz input)
;
; Internal External Switchover:
; IESO_OFF          IESO mode (Two-Speed Start-up) disabled
; IESO_ON           IESO mode (Two-Speed Start-up) enabled

```

----- CONFIG1 (0xabfe) -----

```

; The following settings are available for CONFIG1:
;

```

```

; Watchdog Timer Postscaler:
; WDTPS_PS1        1:1
; WDTPS_PS2        1:2
; WDTPS_PS4        1:4
; WDTPS_PS8        1:8
; WDTPS_PS16       1:16
; WDTPS_PS32       1:32
; WDTPS_PS64       1:64
; WDTPS_PS128      1:128
; WDTPS_PS256      1:256
; WDTPS_PS512      1:512
; WDTPS_PS1024     1:1,024
; WDTPS_PS2048     1:2,048
; WDTPS_PS4096     1:4,096
; WDTPS_PS8192     1:8,192

```

```

;   WDTPS_PS16384      1:16,384
;   WDTPS_PS32768      1:32,768
;
; WDT Prescaler:
;   FWPSA_PR32         Prescaler ratio of 1:32
;   FWPSA_PR128        Prescaler ratio of 1:128
;
; Windowed WDT:
;   WINDIS_ON          Windowed Watchdog Timer enabled; FWDTEN must be 1
;   WINDIS_OFF         Standard Watchdog Timer enabled,(Windowed-mode is disabled)
;
; Watchdog Timer:
;   FWDTEN_OFF         Watchdog Timer is disabled
;   FWDTEN_ON          Watchdog Timer is enabled
;
; Emulator Pin Placement Select bits:
;   ICS_PGx3           Emulator functions are shared with PGEC3/PGED3
;   ICS_PGx2           Emulator functions are shared with PGEC2/PGED2
;   ICS_PGx1           Emulator functions are shared with PGEC1/PGED1
;
; General Segment Write Protect:
;   GWRP_ON            Writes to program memory are disabled
;   GWRP_OFF           Writes to program memory are allowed
;
; General Segment Code Protect:
;   GCP_ON             Code protection is enabled for the entire program memory space
;   GCP_OFF            Code protection is disabled
;
; JTAG Port Enable:
;   JTAGEN_OFF         JTAG port is disabled
;   JTAGEN_ON          JTAG port is enabled

```