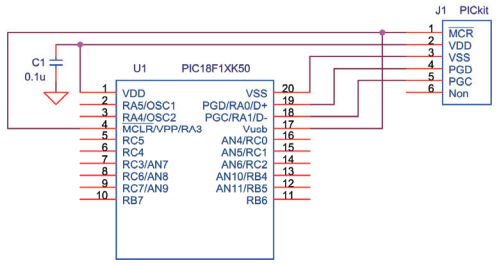
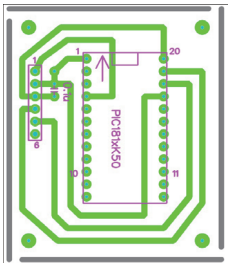
【回路】

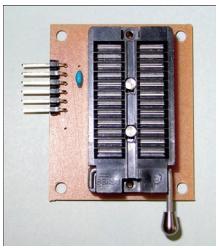
PIC18F40K15　書込みアダプタ

回路図



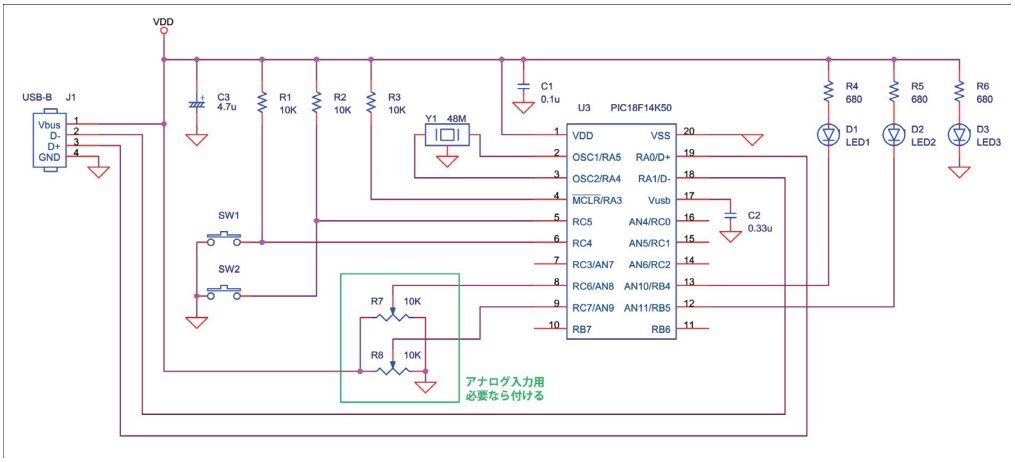
部品面





PIC18F40K15　USBテスト回路

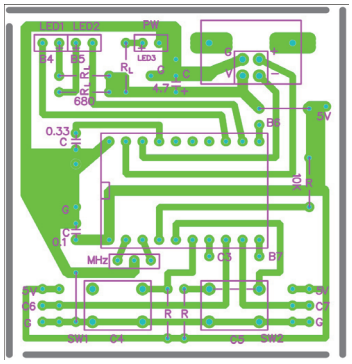
回路図



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

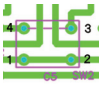


PICのICソケットには、ZIFソケットを使用

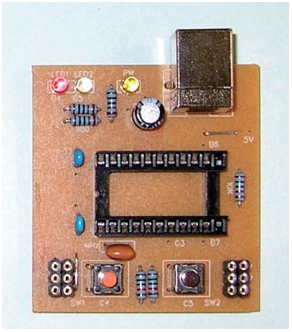
RL：680Ω～1KΩ程度
LEDの電流制限

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。

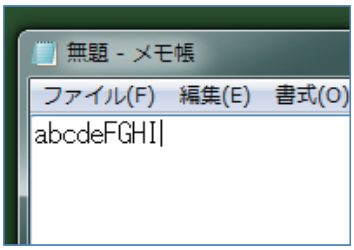
【動作】

1）テスト回路をパソコンに接続すると、LED3が点灯。

2）LED1は常時点灯

キーボードの「Num Lock」キーがONの時、LED2が点灯
キーボードの「Num Lock」キーがOFFの時、LED2が消灯

SW2を押す毎に、パソコンに順に「a」～「0」のキーコードを送る。
SW1を押すと、「Shift」キーのキーコードをパソコンに送る。



【開発】

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
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
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
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_Keyboard」の作成
フォルダ「hid_Keyboard」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 - Keyboard¥Firmware」から、フォルダ「hid_Keyboard」にファイルをコピー
HardwareProfile.h
main.c
rm18f14k50.lkr
usb_config.h
usb_descriptors.c

「C:¥Microchip Solutions v2011-12-05¥Microchip¥USB」から、フォルダ「hid_Keyboard」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_Keyboard¥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_Keyboard」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_Keyboard」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_Keyboard¥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_Keyboard¥USB」にファイルをコピー
usb.h
usb_ch9.h
usb_common.h
usb_device.h
usb_function_hid.h
usb_hal.h
usb_hal_pic18.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_Keyboard」内にプロジェクトファイル「hid_Keyboard.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「hid_Keyboard」

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

Keyboard.c　　ファイルの修正、「Keyboard.c」を参照

rm18f14k50.lkr　　コメントアウトの変更

```
11      //Bootloader
12      //CODEPAGE  NAME=vectors  START=0x0  END=0x29  PROTECTED
13      //CODEPAGE  NAME=bootloader  START=0x2A  END=0xFFF  PROTECTED
14      //CODEPAGE  NAME=page  START=0x1000  END=0x3FFF
15
16      //Application
17      CODEPAGE  NAME=boot  START=0x0  END=0x1F  PROTECTED
18      CODEPAGE  NAME=vectors  START=0x1000  END=0x1029  PROTECTED
19      CODEPAGE  NAME=page  START=0x102A  END=0x3FFF
```

usb_config.h　　コメントアウトの有無、「Keyboard.c」ワークシート参照

```
173     #define USER_SET_REPORT_HANDLER USBHIDCBSetReportHandler
```

または

```
173     //#define USER_SET_REPORT_HANDLER USBHIDCBSetReportHandler
```

制御方式を、ポーリング形式の場合

```
82      #define USB_POLLING
83      //#define USB_INTERRUPT
```

または

制御方式を、割込みイベント形式の場合

```
82      //#define USB_POLLING
83      #define USB_INTERRUPT
```

usb_descriptors.c　　変更なし

usb_device.c　　ファイルパスの変更

```
247     #include "./USB/usb_device_local.h"
```

usb_function_hid.c　　変更なし

Compiler.h　　変更なし

「hid_Keyboard¥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_function_hid.h　　変更なし

usb_hal.h　　変更なし

usb_hal_pic18.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 *********************/
/********************************************************************/

//#define USE_SELF_POWER_SENSE_IO
//#define tris_self_power     TRISAbits.TRISA2    // Input
#if defined(USE_SELF_POWER_SENSE_IO)
#define self_power          PORTAbits.RA2
#else
#define self_power          1
#endif

//#define USE_USB_BUS_SENSE_IO
//#define tris_usb_bus_sense  TRISAbits.TRISA1    // Input
#if defined(USE_USB_BUS_SENSE_IO)
#define USB_BUS_SENSE       PORTAbits.RA1
#else
#define USB_BUS_SENSE       1
#endif


/********************************************************************/
/******** Application specific definitions *************************/
/********************************************************************/

#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

/** LED **/
#define led01               LATBbits.LATB4
#define led02               LATBbits.LATB5


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

#endif
```

【keyboard.c】

・緑色の行は置換え
最初の行から、コメント部分「/******* USB Callback Functions **********/」の前までを、以下の内容に置換える。

・赤色の行について
修正方法１
「usb_config.h」の173行目、「#define USER_SET_REPORT_HANDLER USBHIDCBSetReportHandler」を修正しないなら。
ファイルの最後付近、「// ** USB Class Specific Callback Function(s) ***」以下、青色で示した部分を、コメントに従って修正

修正方法２
「usb_config.h」の173行目、「#define USER_SET_REPORT_HANDLER USBHIDCBSetReportHandler」をコメントアウトして
「//#define USER_SET_REPORT_HANDLER USBHIDCBSetReportHandler」にするなら
ファイルの最後付近、「// ** USB Class Specific Callback Function(s) ***」以下、赤色と青色で示した部分を削除
つまり、「void USBHIDCBSetReportHandler(void)」の関数と「void USBHIDCBSetReportComplete(void)」の関数を削除

```
#ifndef KEYBOARD_C
#define KEYBOARD_C

/** INCLUDES *******************************************************/
#include <p18f14k50.h>
#include "./USB/usb.h"
#include "HardwareProfile.h"
#include "./USB/usb_function_hid.h"

/** CONFIGURATION **************************************************/
// PIC18F14K50
#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

/** VARIABLES ******************************************************/
#pragma udata
BYTE old_sw1,old_sw2;

USB_HANDLE lastINTransmission;
USB_HANDLE lastOUTTransmission;

/** PRIVATE PROTOTYPES *********************************************/

static void InitializeSystem(void);
void ProcessIO(void);
void UserInit(void);
void Keyboard(void);

BOOL Switch1IsPressed(void);
BOOL Switch2IsPressed(void);

void USBCBSendResume(void);
void USBHIDCBSetReportComplete(void);

/** VECTOR REMAPPING ***********************************************/
#if defined(USB_INTERRUPT)
    #pragma udata
    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

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

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

#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)
{
    UserInit();

    #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

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

/*********************************************************************
 * Function:        void UserInit(void)
 ********************************************************************/
void UserInit(void)
{
    /* 入出力ポート設定  */
    TRISA = 0x00;
    TRISB = 0x00;
    TRISC = 0xf0;
    ADCON0bits.ADON = 0;    //AD OFF


    //Initialize all of the PUSH buttons & LED
    mInitAllSwitches();
    old_sw1 = sw1;
    old_sw2 = sw2;

    led01 = 0;    //LED ON
    led02 = 0;

    //initialize the variable holding the handle for the last
    // transmission

    lastINTransmission = 0;
    lastOUTTransmission = 0;
}

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

    //Call the function that behaves like a keyboard
    Keyboard();
```

```
}

void Keyboard(void)
{
    static unsigned char key = 4;

    //Check if the IN endpoint is not busy, and if it isn't check if we want to send
    //keystroke data to the host.
    if(!HIDTxHandleBusy(lastINTransmission))
    {
        if(Switch2IsPressed())    //SW2 ON?
        {
            //Load the HID buffer
            hid_report_in[0] = 0;    //R-G,A,S,C, L-G,A,S,C

            if ( sw1 == 0 ) {    //SW1 ON?
                hid_report_in[0] = 0x02;    //R-G,A,S,C, L-G,A,S,C
            }

            hid_report_in[1] = 0;       //0x00
            hid_report_in[2] = key++;   //key 1
            hid_report_in[3] = 0;       //key 2
            hid_report_in[4] = 0;       //key 3
            hid_report_in[5] = 0;       //key 4
            hid_report_in[6] = 0;       //key 5
            hid_report_in[7] = 0;       //key 6

            //Send the 8 byte packet over USB to the host.
            lastINTransmission = HIDTxPacket(HID_EP, (BYTE*)hid_report_in, 0x08);

            if(key == 40)
            {
                key = 4;
            }
        }
        else
        {
            //Load the HID buffer
            hid_report_in[0] = 0;
            hid_report_in[1] = 0;
            hid_report_in[2] = 0;   //Indicate no character pressed
            hid_report_in[3] = 0;
            hid_report_in[4] = 0;
            hid_report_in[5] = 0;
            hid_report_in[6] = 0;
            hid_report_in[7] = 0;
            //Send the 8 byte packet over USB to the host.
            lastINTransmission = HIDTxPacket(HID_EP, (BYTE*)hid_report_in, 0x08);
        }
    }


    //Check if any data was sent from the PC to the keyboard device.  Report descriptor allows
    //host to send 1 byte of data.  Bits 0-4 are LED states, bits 5-7 are unused pad bits.
    //The host can potentially send this OUT report data through the HID OUT endpoint (EP1 OUT),
    //or, alternatively, the host may try to send LED state information by sending a
    //SET_REPORT control transfer on EP0.  See the USBHIDCBSetReportHandler() function.
    if(!HIDRxHandleBusy(lastOUTTransmission))
    {
        lastOUTTransmission = HIDRxPacket(HID_EP,(BYTE*)&hid_report_out,1); //Data is in the OutBuffer[0].

        //Num Lock LED state is in Bit0.
        if(hid_report_out[0] & 0x01) //Make LED1 and LED2 match Num Lock state.
        {
            led02 = 0;        //LED2 On;
        }
        else
        {
            led02 = 1;        //LED2 Off
        }
    }

    return;
}


/********************************************************************************
 * 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 **********************************************************
// ************************************************************************************************
```

```
// ******************************************************************************
// ************** USB Class Specific Callback Function(s) **********************
// ******************************************************************************

/********************************************************************
 * Function:        void USBHIDCBSetReportHandler(void)
 *
 * PreCondition:    None
 *
 * Input:           None
 *
 * Output:          None
 *
 * Side Effects:    None
 *
 * Overview:        USBHIDCBSetReportHandler() is used to respond to
 *                    the HID device class specific SET_REPORT control
 *                    transfer request (starts with SETUP packet on EP0 OUT).
 * Note:
 ********************************************************************/
void USBHIDCBSetReportHandler(void)
{
    //Prepare to receive the keyboard LED state data through a SET_REPORT
    //control transfer on endpoint 0.  The host should only send 1 byte,
    //since this is all that the report descriptor allows it to send.
    USBEP0Receive((BYTE*)&CtrlTrfData, USB_EP0_BUFF_SIZE, USBHIDCBSetReportComplete);
}

//Secondary callback function that gets called when the above
//control transfer completes for the USBHIDCBSetReportHandler()
void USBHIDCBSetReportComplete(void)
{
    //1 byte of LED state data should now be in the CtrlTrfData buffer.

    //Num Lock LED state is in Bit0.
    if(CtrlTrfData[0] & 0x01)    //Make LED1 and LED2 match Num Lock state.
    {
        mLED_1_On();   //修正すること、この行と次の行を、led01 = 0; の１行に変更
        mLED_2_On();
    }
    else
    {
        mLED_1_Off();  //修正すること、この行と次の行を、led01 = 1; の１行に変更
        mLED_2_Off();
    }

    //Stop toggling the LEDs, so you can temporily see the Num lock LED state instead.
    //Once the CountdownTimerToShowUSBStatusOnLEDs reaches 0, the LEDs will go back to showing USB state instead.
    BlinkStatusValid = FALSE;
    CountdownTimerToShowUSBStatusOnLEDs = 140000;
}


/** EOF Keyboard.c ***************************************************/
#endif
```

【備考、KeyCode】

keyboard input report (8bytes)

Byte	Description
0	Modifier keys
1	Reserved
2	Keycode(1)
3	Keycode(2)
4	Keycode(3)
5	Keycode(4)
6	Keycode(5)
7	Keycode(6)

keyboard output report (1byte)

Bit	Description
0	NUM LOCK
1	CAPS LOCK
2	SCROLL LOCK
3	COMPOSE
4	KANA
5 to 7	CONSTANT

同時に送れるキーコードは6個

Modifier keys

Bit	
0	LEFT CTRL
1	LEFT SHIFT
2	LEFT ALT
3	LEFT GUI
4	RIGHT CTRL
5	RIGHT SHIFT
6	RIGHT ALT
7	RIGHT GUI

Keycode

ID (Dec)	ID (Hex)	Name
4	H04	a
5	H05	b
6	H06	c
7	H07	d
8	H08	e
9	H09	f
10	H0A	g
11	H0B	h
12	H0C	i
13	H0D	j
14	H0E	k
15	H0F	l
16	H10	m
17	H11	n
18	H12	o
19	H13	p
20	H14	q
21	H15	r
22	H16	s
23	H17	t
24	H18	u
25	H19	v
26	H1A	w
27	H1B	x
28	H1C	y
29	H1D	z
30	H1E	1
31	H1F	2
32	H20	3
33	H21	4
34	H22	5
35	H23	6
36	H24	7
37	H25	8
38	H26	9
39	H27	0

【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
;
;==========================================================================
```