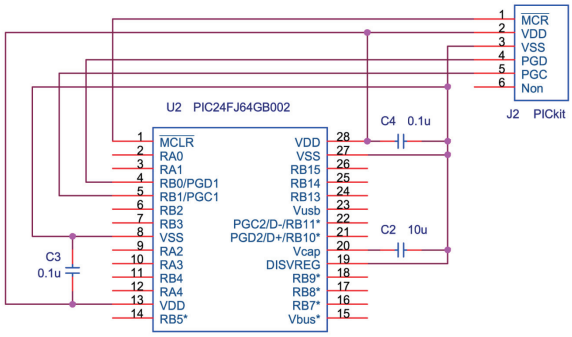
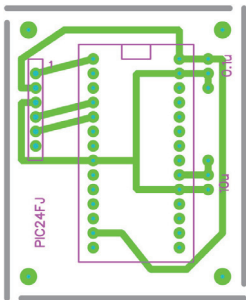
【回路図】

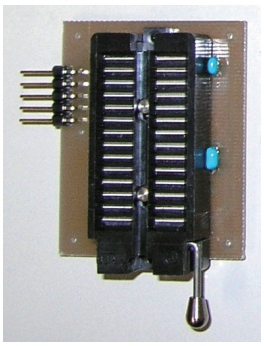
PIC24F64Gx002　書込みアダプタ

回路図



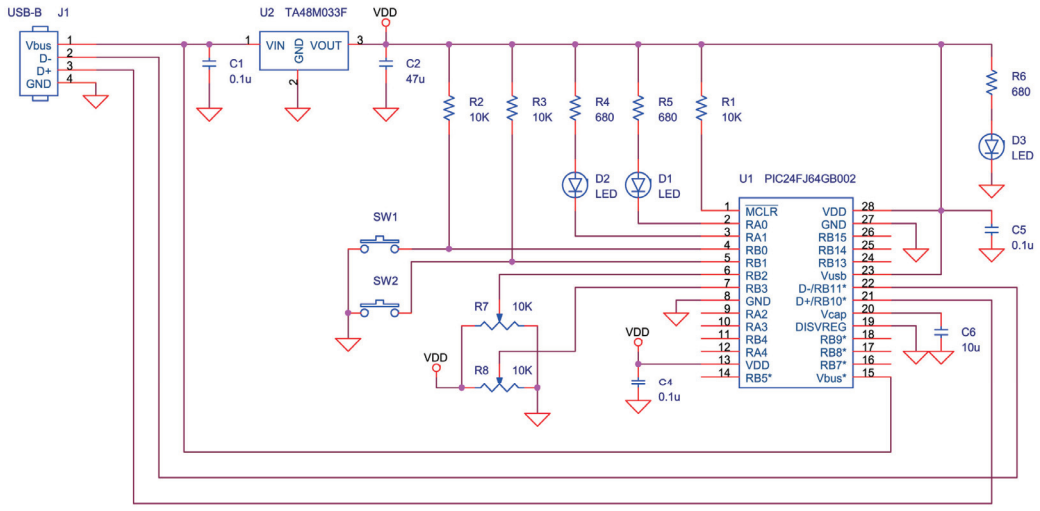
部品面





PIC24F64GB002　USBテスト回路

回路図

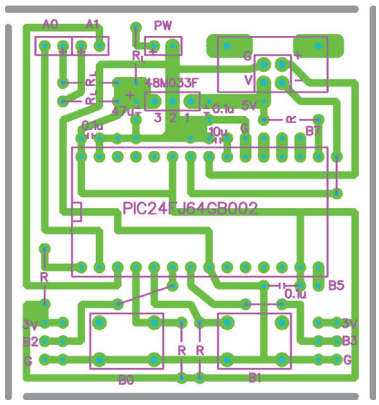


記号	名称	個数
USB-B	USBコネクタBタイプ	1
C4,C5	セラミックコンデンサ　0.1μF	1
C6	セラミックコンデンサ　10μF	1
R1～R3	抵抗　10KΩ	2
R4～R6	抵抗　680Ω～1KΩ	3
R7,R8	可変抵抗　5KΩ～10KΩ	2
D1～D3	LED	3
SW1～SW2	押しボタンスイッチ 押したときだけON	2
U1	ゼロプレッシャー（ZIF）	1
	ICソケット	1
	PIC24F64GB002	1
U2	5V　->　3.3V　レギュレータ	1
C1	セラミックコンデンサ　0.1μF	1
C2	電界コンデンサ　47μF	1

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

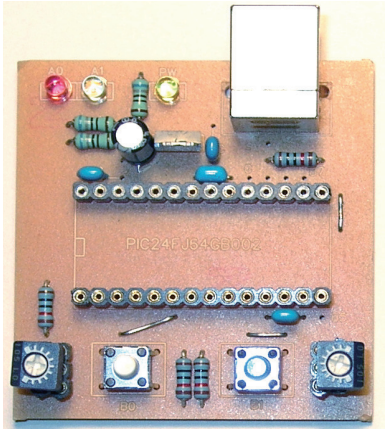


PICのICソケットには、ZIFソケットを使用

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：RB0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。

Copyright(C) 2010 NIPS Technical Division. All Rights Reserved.

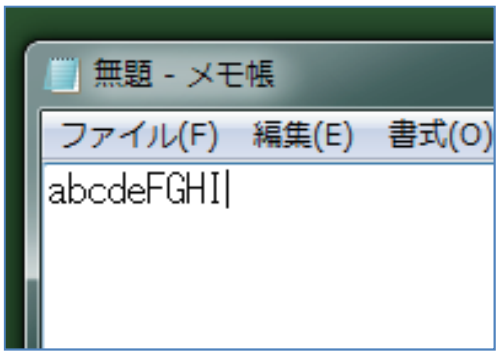
【動作】

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。



Copyright(C) 2010 NIPS Technical Division. All Rights Reserved.

【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
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
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
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。

Copyright(C) 2010 NIPS Technical Division. All Rights Reserved.

【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_Joystick」を作成
フォルダ「hid_Joystick」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 - Keyboard¥Firmware」から、フォルダ「hid_Keyboard」にファイルをコピー
HardwareProfile.h
main.c　　<-ファイル名の変更、「main.c」を「keyboard.c」に変更する
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_device_local.h
usb_hal_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
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_pic24.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_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_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

Copyright(C) 2010 NIPS Technical Division. All Rights Reserved.

【ファイルの修正】

ファイル名　　行番号　修正および追加内容

「hid_Joystick」ディレクトリ内

HardwareProfile.h　　行の置換え
全部　HardwareProfile.hの修正を参照

Keyboard.c　　行の置換え
前半部　Joystick.cの修正を参照
後半部　必要に応じて変更、Joystick.cの修正を参照

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　　ディレクトリパスの修正
247　#include "./USB/usb_device_local.h"

「hid_Joystick¥USB」ディレクトリ内

usb.h　　ディレクトリパスの修正

```
110    #include "../GenericTypeDefs.h"
111    #include "../Compiler.h"
112
113    #include "../usb_config.h"            // Must be defined by the application
114
115    #include "../USB/usb_common.h"        // Common USB library definitions
116    #include "../USB/usb_ch9.h"           // USB device framework definitions
117
118    #if defined( USB_SUPPORT_DEVICE )
119        #include "../USB/usb_device.h"    // USB Device abstraction layer interf
120    #endif
121
122    #if defined( USB_SUPPORT_HOST )
123        #include "../USB/usb_host.h"      // USB Host abstraction layer interfac
124    #endif
125
126    #if defined ( USB_SUPPORT_OTG )
127        #include "USB/usb_otg.h"
128    #endif
129
130    #include "../USB/usb_hal.h"           // Hardware Abstraction Layer interfac
```

usb_hal.h　　ディレクトリパスの修正
101　#include "../USB/usb_hal_pic24.h"

usb_hal_pic24.h　　ディレクトリパスの修正
115　#include "../Compiler.h"
116　#include "../usb_config.h"

usb_device_local.h　　ディレクトリパスの修正
88　#include "../usb_config.h"

Copyright(C) 2010 NIPS Technical Division. All Rights Reserved.

【HardwareProfile.hの修正】

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
#define self_power          1

//#define USE_USB_BUS_SENSE_IO
//#define tris_usb_bus_sense  U1OTGSTATbits.SESVD  //TRISBbits.TRISB5    // Input
#define USB_BUS_SENSE       1


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

#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

/** LED **/
#define led01               LATAbits.LATA0
#define led02               LATAbits.LATA1

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

#endif  //HARDWARE_PROFILE_H
```

【Keyboard.cの修正】

・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 <p24fj64gb002.h>
#include "./USB/usb.h"
#include "HardwareProfile.h"
#include "./USB/usb_function_hid.h"

/** CONFIGURATION **************************************************/
// 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)

/** VARIABLES ******************************************************/
#pragma udata
BYTE old_sw1,old_sw2;
char buffer[8];
unsigned char OutBuffer[8];

#pragma udata
USB_HANDLE lastINTransmission;
USB_HANDLE lastOUTTransmission;


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

void USBHIDCBSetReportComplete(void);

/** VECTOR REMAPPING ***********************************************/


/** 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)
{
    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・ソフトウエア OFF
    AD1PCFG = 0xFFFF;                   //AD OFF

    /* 入出力ポート設定  */
    TRISA = 0x0000;                     //ポートA、全て出力
    TRISB = 0x0003;                     //ポートB、Bit0-1入力、Bit2-7出力


    //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、キーコード例】

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】

「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:
;     WPFP_WPFP0            Page 0 (0x0)
;     WPFP_WPFP1            Page 1 (0x400)
;     WPFP_WPFP2            Page 2 (0x800)
;     WPFP_WPFP3            Page 3 (0xC00)
;     WPFP_WPFP4            Page 4 (0x1000)
;     WPFP_WPFP5            Page 5 (0x1400)
;     WPFP_WPFP6            Page 6 (0x1800)
;     WPFP_WPFP7            Page 7 (0x1C00)
;     WPFP_WPFP8            Page 8 (0x2000)
;     WPFP_WPFP9            Page 9 (0x2400)
;     WPFP_WPFP10           Page 10 (0x2800)
;     WPFP_WPFP11           Page 11 (0x2C00)
;     WPFP_WPFP12           Page 12 (0x3000)
;     WPFP_WPFP13           Page 13 (0x3400)
;     WPFP_WPFP14           Page 14 (0x3800)
```

```
;     WPFP_WPFP15          Page 15 (0x3C00)
;     WPFP_WPFP16          Page 16 (0x4000)
;     WPFP_WPFP17          Page 17 (0x4400)
;     WPFP_WPFP18          Page 18 (0x4800)
;     WPFP_WPFP19          Page 19 (0x4C00)
;     WPFP_WPFP20          Page 20 (0x5000)
;     WPFP_WPFP21          Page 21 (0x5400)
;     WPFP_WPFP22          Page 22 (0x5800)
;     WPFP_WPFP23          Page 23 (0x5C00)
;     WPFP_WPFP24          Page 24 (0x6000)
;     WPFP_WPFP25          Page 25 (0x6400)
;     WPFP_WPFP26          Page 26 (0x6800)
;     WPFP_WPFP27          Page 27 (0x6C00)
;     WPFP_WPFP28          Page 28 (0x7000)
;     WPFP_WPFP29          Page 29 (0x7400)
;     WPFP_WPFP30          Page 30 (0x7800)
;     WPFP_WPFP31          Page 31 (0x7C00)
;     WPFP_WPFP32          Page 32 (0x8000)
;     WPFP_WPFP33          Page 33 (0x8400)
;     WPFP_WPFP34          Page 34 (0x8800)
;     WPFP_WPFP35          Page 35 (0x8C00)
;     WPFP_WPFP36          Page 36 (0x9000)
;     WPFP_WPFP37          Page 37 (0x9400)
;     WPFP_WPFP38          Page 38 (0x9800)
;     WPFP_WPFP39          Page 39 (0x9C00)
;     WPFP_WPFP40          Page 40 (0xA000)
;     WPFP_WPFP41          Page 41 (0xA400)
;     WPFP_WPFP42          Page 42 (0xA800)
;     WPFP_WPFP63          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_WPCFGEN        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 WPFP
;     WPEND_WPENDMEM       Write Protect from WPFP 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 PLLEN 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
```