FRマイコン基板を使ったUSBホスト・システムの設計事例 後編

USBジョイスティックをつなげてみよう！

関連データ

根岸 智明

本誌2008年5月号付属FRマイコン基板の特徴として，USBホスト＆ターゲット機能を内蔵している点が挙げられる．FRマイコン基板のUSBホスト応用事例として，ここではUSBジョイスティックを接続する事例について解説する．USBホスト・スタックにはmini-USBHostSTRを使用している．（編集部）

● MB91FV310AのUSBホスト機能の活用

先月号（2008年7月号，pp.148-151）に引き続き，本誌2008年5月号付属FRマイコン基板（**写真1**）を，さまざまなUSB周辺機器を接続するための変換アダプタとして使う事例について解説します．今回はUSBジョイスティックを接続し，方向ボタンまたはスティックを倒した状態などを簡単に判定できるアプリケーションを作成します．

なお今回も，USBホスト用プロトコル・スタックとしてmini-USBHostSTR（インターフェイス社製）を使用しています．mini-USBHostSTRについての詳細は本誌2008年6月号を参照してください．

● サンプル・プログラムの動作仕様

本誌付属FRマイコン基板を使用し，USBジョイスティックに対応したソフトウェアの動作仕様を，実際のソース・ファイルを使って説明します．

USBジョイスティックに対応したのアプリケーションのソース・ファイルを**リスト1**に示します．リストを見ると分かるように，`main()`関数そのものは非常にシンプルな作りとなっています．実際にUSBジョイスティックを動作（ログ出力）させるために使用しているAPIは，**表1**と**表2**に示す二つのみです．

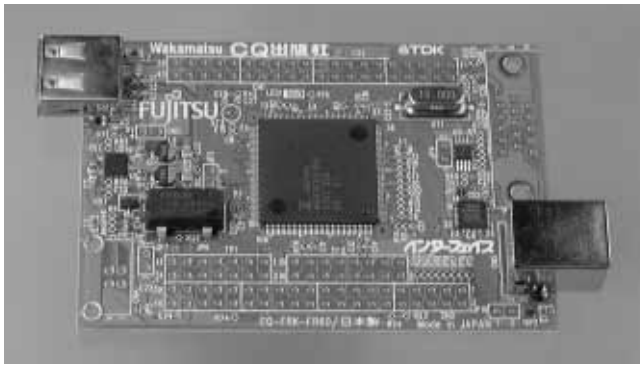

写真1　本誌5月号付属FRマイコン基板

● APIの動作

この二つのAPIには，それぞれコールバック関数を指定することができます．また，`Usb_Host_Init()`に指定したコールバック関数〔**リスト1**では`Usb_Host_Callback()`がこれに相当〕は，デバイスの状態変化（接続，切断など）に合わせて呼び出されます．さらに`Usb_Hid_Joystick_Setting()`に指定したコールバック関数〔**リスト1**では`Usb_Hid_Joystick_Event()`がこれに相当〕は，USBジョイスティックの状態の変化に合わせて呼び出されます．

以上のように，状態の変化に合わせてUSBホスト・スタックから呼び出されるコールバック関数を用い，ログ出力（FRマイコン基板のCN5）を行うことでソフトウェアを実現しています．

`Usb_Host_Callback()`関数を見ると，デバイスの状態変化に合わせて，ログ（“CN:USB JOYSTICK”，“CN:Unknown”，“DS:”）を出力しているのが分かると思います．

また，`Usb_Hid_Joystick_Event()`関数では，USB

表1　USBホストスタックのAPI

	型	内　容
引き数	なし	コールバック関数
戻り値	`ISTATUS(long)`	0：正常終了，負数：異常終了

（**a**）USBホスト・スタック初期化関数`Usb_Host_Init()`

	型	内　容
引き数	`USBH_JOYSTICK_SETTING_T`	設定データ（コールバック関数）
戻り値	`ISTATUS(long)`	0：正常終了，負数：異常終了

（**b**）USBジョイスティック・クラス・ドライバへの設定`Usb_Hid_Joystick_Setting()`

表2　USBジョイスティック・クラス・ドライバへの設定データ構造体（`USBH_JOYSTICK_SETTING_T`）

フィールド名	内　容
`void(*eventf)(IHANDLE, int32_t, void *)`	コールバック関数

リスト1　ジョイスティック対応のアプリケーションのソース・ファイル

```
/*********************************************************************/
/**
 * mini-USBHost Joystick DEMOアプリケーション.
 * Copyright (c) 2008 Interface Inc. All Rights reserved.
 */
/*********************************************************************/

/*-------------------------------------------*/
/*                  インクルード             */
/*-------------------------------------------*/
#include "itflib.h"
#include "miniusbhost.h"
#include "serial.h"
#include "timer.h"


/*-------------------------------------------*/
/*                  宣言                     */
/*-------------------------------------------*/
#define MSG(a)
{Serial_Send((uint8_t*)(a), strlen(a));}
#define MSG1(a,b)
{sprintf(g_dbg_buffer, a, b);MSG(g_dbg_buffer);}

/*-------------------------------------------*/
/*                  内部変数                 */
/*-------------------------------------------*/
char      g_dbg_buffer[256];

/*-------------------------------------------*/
/*                  内部関数                 */
/*-------------------------------------------*/
static void Usb_Hid_Joystick_Event(IHANDLE devhdl, void *prm);


/*********************************************************************/
/**
 * Hostコールバック関数.
 *
 * @param devhdl [in]      デバイス・ハンドル
 * @param event [in]       イベント・コード
 * @param prm [in]         パラメータ
 */
void Usb_Host_Callback(IHANDLE devhdl, int32_t event, void *prm)
{
    union USBH_CALLBACK_PRM_T *info = (union USBH_CALLBACK_PRM_T *)prm;

    switch (event) {
    case USBH_EVENT_CONNECT:
         switch (info->devinfo.class_type) {
         case USB_HOST_CLASS_HID_INPUT:
/* Human Interface Device Class */
              MSG("CN:USB JOYSTICK\r\n");
              break;
         default:
              MSG("CN:Unknown\r\n");
              break;
         }
         break;
    case USBH_EVENT_DISCONNECT:
         MSG("DS:\r\n");
         break;
    default:
         break;
    }
}

/*********************************************************************/
/**
 * mini-USBHost HID DEMOアプリケーション.
 */
void main(void)
{
    ISTATUS result;
    USBH_JOYSTICK_SETTING_T joystickSetting = {Usb_Hid_Joystick_Event};

    Serial_Init();
    Timer_Init(__USBH_INTERVAL_TIMER__, Usb_Host_Timer);

    result = Usb_Host_Init(Usb_Host_Callback);
    if (result) {
         for (;;);
    }

    result = Usb_Hid_Joystick_Setting(&joystickSetting);
    if (result) {
         for (;;);
    }

    Timer_Start();

    while (TRUE) {
         ;
    }
}

/*********************************************************************/
/**
 * HID Joystick イベント・ルーチン.
 *
 * @param devhdl [in]      デバイス・ハンドル
 * @param prm [in]         パラメータ
 */
static void Usb_Hid_Joystick_Event(IHANDLE devhdl, void *prm)
{
    USBH_HID_JOYSTICK_T *pjs = (USBH_HID_JOYSTICK_T *)prm;
    int32_t i;

    MSG("\r\nAxis       : ");
    if (pjs->info.X.Enable) {
         MSG1("X:%d ", pjs->state.X);
    }
    if (pjs->info.Y.Enable) {
         MSG1("Y:%d ", pjs->state.Y);
    }
    if (pjs->info.Z.Enable) {
         MSG1("Z:%d ", pjs->state.Z);
    }
    if (pjs->info.Rx.Enable) {
         MSG1("Rx:%d ", pjs->state.Rx);
    }
    if (pjs->info.Ry.Enable) {
         MSG1("Ry:%d ", pjs->state.Ry);
    }
    if (pjs->info.Rz.Enable) {
         MSG1("Rz:%d ", pjs->state.Rz);
    }

    if (pjs->info.MaxHatSwitch) {
         MSG("\r\nHat Switch : ");
         for (i = 0; i < pjs->info.MaxHatSwitch; i++) {
              MSG1("%d ", pjs->state.HatSwitch[i]);
         }
    }

    if (pjs->info.MaxButton) {
         MSG("\r\nButton     : ");
         for (i = 0; i < pjs->info.MaxButton; i++) {
              MSG1("%d ", pjs->state.Button[i]);
         }
    }

    MSG("\r\n");
}
```