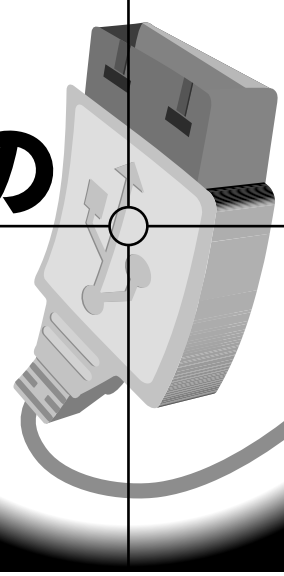
FRマイコン基板を使った

USBターゲット・システムの設計事例

USBマウス・エミュレーション・プログラムの作成



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ターゲットの応用事例として，USBマウスをエミュレーションするサンプル・プログラムを作成する．ここで作成したUSBターゲットについては，ソース・コードをすべて公開する．（編集部）

末武 清次

● USBマウス・エミュレーションとは

MB91FV310Aに搭載しているUSBターゲット機能を使用した例として，USBマウスをエミュレーションするプログラムを作成しました．OS標準のHIDマウス・ドライバで動作するため，ホスト側に特別なデバイス・ドライバなどは不要です．FRマイコン基板のCN4をパソコンに接続するだけでOS標準のマウス・ドライバが組み込まれ，マウス・カーソルが勝手に動き回るというものです．

● MB91FV310AのUSBターゲットの仕様

MB91FV310Aに搭載されているUSBターゲット・コントローラには，コントロール転送，バルクIN転送，バルクOUT転送，インタラプトIN転送の各エンドポイントがあり，それぞれ8バイト，64バイト，64バイト，8バイトのバッファ・サイズとなります(**表1**)．バルク・エンドポイントはダブル・バッファ構成で，DMA転送にも対応しています．

搭載されているUSBプロトコル・エンジンは，標準リクエストに関して，

- GET_DESCRIPTOR
- SET_DESCRIPTOR
- SYNC_FRAME

以外は自動応答を行います．上記三つの標準リクエストの処理と，クラス・リクエストやベンダ・リクエストはソフトウェアで処理する必要があります．

● 外部バス・インターフェースと外部割り込みを使う

MB91FV310Aでは，USBターゲット・コントローラやUSBホスト・コントローラ，さらにOSDコントローラまでも外部バス・インターフェースを介して接続されており，割り込みは外部割り込みのチャネル4～6につながっています．そのため，外部バス・インターフェースと外部割り込みの初期化が必要です．詳細については先月号の特集Appendix(2008年6月号，pp.66-67)を参照してください．

外部バス・インターフェースの初期化で，サンプルのスケルトン・プロジェクトを使用する際は，モード・ベクタ定義を，

```
#define MODV    0x05000000
```

と変更してください．

外部割り込みは，検出レベルを“H”レベル検出にし，要因ビットをクリアした後に割り込み許可にします．また，割り込みレベル設定ICRレジスタにもレベルを設定しておきます．

● D+端子のプルアップ処理

MB91FV310AのUSBターゲットはフルスピード・モード対応なので，ホストと通信する際はD+端子をプルアップします．付属FRマイコン基板でのプルアップ制御は，汎用ポートのP36で行います．PDR3レジスタのビット6を1にしてプルアップをOFFに，DDR3レジスタのビット6を1にして出力方向に設定します．

USBターゲットの初期化が終わってから，PDR3レジスタのビット6を0にしてプルアップをONにします．

● USBターゲットの初期化

USBターゲット・コントローラの初期化は`usbfu.c`(リ

表1　マウス・エミュレート時のエンドポイントの構成

エンドポイント	コンフィグレーション	インターフェース	オルタネート	転送タイプ	最大パケット・サイズ(バイト)
0	—	—	—	コントロール	8
1	1	0	0	バルクOUT	64
2	1	0	0	バルクIN	64
3	1	0	0	インタラプトIN	8

KEYWORD——MB91FV310A，USBターゲット，ディスクリプタ，USBマウス

スト1）内のUSB_initで行っています．まず同期リセット処理を行います．0x0006FFFE番地のマクロ・リセット・レジスタでUSB-F-RSTを1にセットして，48MHzで16サイクルの間リセットします．サンプル・プログラムではCPUがNOP命令を16回実行するのを待ってからリセット解除を行っています．

その次にエンドポイントの初期化を行います．初期化用データをFIFO2に書き込んでCFGENを1にセットし，CFENDが立つまで待ちます．

以降は，各設定レジスタを初期化します．FIFOのクリアやSTALL解除，割り込みマスクの設定などを行います．

初期化が完了したら，D+端子のプルアップをONにして，USBホストにデバイスの接続を通知します．

● 割り込み処理

usbfu.c内のINT4_int割り込み処理関数で割り込み処理を行っています．サンプル・プログラムでは，次の割り込みを処理しています．

- エンドポイント0のIN転送ACK/NACKとOUT転送ACK/NACK
- エンドポイント3のACK/NACK
- USBバス・リセット

USBホストがデバイス接続を認識するとUSBバス・リセットが発生します（**図1**）．サンプル・プログラムでは，バス・リセットを検出したら，エンドポイントFIFOをクリアするようにしています（115～117行目）．割り込みルーチンの最後で，USBの割り込み要因ビットだけでなく，外部割り込みの要因ビットも忘れずにクリアします．

● コントロール転送

バス・リセット後は，USBターゲット認識のためのコントロール転送が行われます．エンドポイント0のOUT転送に対するACK割り込み（ACK0o）が発生したらEP0_out関数を呼び出します．EP0_outでは，受信データ・

リスト1　USBマウス・エミュレーションのUSBターゲット・コントローラ制御部（usbfu.c）

```
001: /* MB91FV310A USB-Function Sample */
002:
003: #include "_fr.h"
004: #include "extern.h"
005: #include "usbfu.h"
006: #include "usb_descripter.h"
007:
008: USBDATA CTRL_OUT[4];/* CONTROL-OUT data buffer(8byte) */
009: USB_SEND_BUF usb_send_buffer;
010:
011: unsigned int cnt = 0;
012:
013: void USB_init(void)
014: {
015:     unsigned int i;
016:     volatile unsigned char *IPRST;
017:
018:     IPRST = (unsigned char *)0x0006FFFE;
019:     *IPRST = 0x40; /* USB-F-RST assert */
020:     for(i=0;i<16;i++) __wait_nop();
021:     *IPRST = 0x00; /* USB-F-RST negate */
022:
023:     /* Endpoint Initialize */
024:     IO_USBF.IO_FIFO2 = 0x0000;
025:     IO_USBF.IO_FIFO2 = 0x1080;
026:     IO_USBF.IO_FIFO2 = 0x0014;
027:     IO_USBF.IO_FIFO2 = 0x2080;
028:     IO_USBF.IO_FIFO2 = 0x8001;
029:     IO_USBF.IO_FIFO2 = 0x2428;
030:     IO_USBF.IO_FIFO2 = 0x8080;
031:     IO_USBF.IO_FIFO2 = 0x0234;
032:     IO_USBF.IO_FIFO2 = 0x3810;
033:     IO_USBF.IO_FIFO2 = 0x8003;
034:
035:     IO_USBF.IO_CONT1.bit.CFGEN = 1;
036:
037:     while(!IO_USBF.IO_ST3.bit.CFEND);
038:
039:     /* FIFO clear */
040:     IO_USBF.IO_CONT2.bit.INI0o = 1;
041:     IO_USBF.IO_CONT2.bit.INI0i = 1;
042:     IO_USBF.IO_CONT2.bit.INI1 = 1;
043:     IO_USBF.IO_CONT2.bit.INI2 = 1;
044:     IO_USBF.IO_CONT2.bit.INI3 = 1;
045:
046:     /* ACK IRQ MASK */
047:     IO_USBF.IO_CONT7.bit.MACK0o = 1;
048:     IO_USBF.IO_CONT7.bit.MACK0i = 1;
049:     IO_USBF.IO_CONT7.bit.MACK1 = 0;
050:     IO_USBF.IO_CONT7.bit.MACK2 = 0;
051:     IO_USBF.IO_CONT7.bit.MACK3 = 1;
052:
053:     /* NACK IRQ MASK */
054:     IO_USBF.IO_CONT8.bit.MNACK0o = 1;
055:     IO_USBF.IO_CONT8.bit.MNACK0i = 1;
056:     IO_USBF.IO_CONT8.bit.MNACK1 = 0;
057:     IO_USBF.IO_CONT8.bit.MNACK2 = 0;
058:     IO_USBF.IO_CONT8.bit.MNACK3 = 1;
059:
060:     /* DMA request Disable */
061:     IO_USBF.IO_CONT5.bit.DFIFOBUSY2 = 0;
062:     IO_USBF.IO_CONT5.bit.DFIFOBUSY1 = 0;
063:
064:     /* FIFO BUSY clear */
065:     IO_USBF.IO_CONT4.bit.FIFOBUSY0o = 1;
066:     IO_USBF.IO_CONT4.bit.FIFOBUSY0i = 1;
067:     IO_USBF.IO_CONT4.bit.FIFOBUSY1 = 0;
068:     IO_USBF.IO_CONT4.bit.FIFOBUSY2 = 0;
069:     IO_USBF.IO_CONT4.bit.FIFOBUSY3 = 1;
070:
071:     /* Interrupt MASK */
072:     IO_USBF.IO_CONT9.bit.MSTALL0 = 0;
073:     IO_USBF.IO_CONT9.bit.MSTALL1 = 0;
074:     IO_USBF.IO_CONT9.bit.MSTALL2 = 0;
075:     IO_USBF.IO_CONT9.bit.MSTALL3 = 0;
076:     IO_USBF.IO_CONT9.bit.MUSBRESET = 1;
077:
078:     IO_USBF.IO_CONT3.bit.BFOK0o = 1;
/* Endpoint0 out Enable */
079:     IO_USBF.IO_CONT3.bit.BFOK0i = 1;
/* Endpoint0 in Enable */
080:     IO_USBF.IO_CONT3.bit.BFOK3 = 1;
/* Endpoint3 in Enable */
081: }
082:
083: /* INT4/USB interrupt routine */
084: __interrupt void INT4_int(void)   ← 割り込み処理関数
085: {
```