

SPI/USB 変換ユニット

外部仕様書

第 3 版

(2010 / 07 / 22)

はじめに

C Dの内容

外部仕様書(第3版).pdf (本ドキュメントファイル)

¥driver フォルダ

F2Usb.sys : SPI/USB 変換ユニット専用USBドライバファイル

F2Usb.inf : USBドライバインストール用INFファイル

¥dll フォルダ

F2Usb.dll : USBドライバアクセス用DLLファイル

F2Usb.h : DLLアクセス用ヘッダファイル(アプリケーション作成用)

Sample.exe : サンプルアプリケーション

※ **Sample.exe**は、ご使用環境でのUSBドライバの動作対応確認用です。本仕様で定義しているUSB通信パケット処理などの機能を実装したものではありません。

USBドライバの動作環境

- PC : USBコネクタを備えているもの
- OS : Microsoft®Windows®2000
Microsoft®Windows®XP(SP2)
Microsoft®Windows Vista®
Microsoft®Windows7®

※ 弊社で動作確認の取れているOSを掲載しています。

PCアプリケーションの開発環境について

F2Usb.h、**F2Usb.dll** を利用してPCアプリケーションを開発するためには、

ヘッダファイル : <setupapi.h>

キーワード : WINAPI, DWORD, __declspec(dllexport), __stdcall

に対応したC/C++言語互換環境が必要です。

なお、弊社でコンパイル確認の取れている開発環境は、以下の通りです。

- Boland C++Builder6
- Microsoft Visual C++

Microsoft、**Windows**、**Windows Vista**、および **Windows7** は米国 **Microsoft Corporation** の米国およびその他の国における登録商標または商標です。

その他、本書に記載されている現れる社名、製品名は各社の登録商標または商標です。

目次

| | | |
|------|--|----|
| 1. | SPI/USB 変換ユニット本体の DIP-SW 設定 | 1 |
| 2. | SPI/USB 変換ユニット本体の LED 表示 | 1 |
| 3. | ホスト P C との接続 | 2 |
| 4. | P C アプリケーションの開発 | 3 |
| 4. 1 | DLL の利用 | 3 |
| 4. 2 | U S B インタフェース仕様 | 3 |
| 4. 3 | U S B 通信パケット | 4 |
| 4. 4 | レジスタマップ | 5 |
| 4. 5 | U S B 通信フロー | 7 |
| | (1) 接続確認処理 | 7 |
| | (2) ホスト P C からゲーム機へのデータ転送手順（600 バイト転送の例） | 8 |
| | (3) ゲーム機からホスト P C へのデータ転送手順（800 バイト転送の例） | 9 |
| 4. 6 | 通信所要時間の評価 | 10 |
| | (1) full speed 接続：U=63 の場合 | 10 |
| | (2) high speed 接続：U=511 の場合 | 11 |

表目次

| | | |
|-------|--|----|
| 表 1 : | D I P - S W と動作モードの関係 | 1 |
| 表 2 : | L E D 表示パターン | 1 |
| 表 3 : | 変換ユニットの U S B 通信基本仕様 | 3 |
| 表 4 : | 制御コマンドパケット | 4 |
| 表 5 : | データパケット | 4 |
| 表 6 : | 書き込み完了通知パケット | 4 |
| 表 7 : | 変換ユニットの制御用レジスタ | 5 |
| 表 8 : | $L \leq 4096$ の場合のスループット見込み値（full speed 接続時） | 10 |
| 表 9 : | U S B フレームスケジューリング実測平均値（full speed 接続時） | 10 |

1．SPI/USB 変換ユニット本体の DIP-SW 設定

SPI/USB 変換ユニット（以下、変換ユニット）の DIP-SW は、下表の各動作モードに対応しています。

※ DIP-SW は変換ユニット起動前に設定して下さい。動作中に DIP-SW を操作しても無視されます。

※ 変換ユニットに DIP-SW が実装されていない場合（初回納品分）は、全ての DIP-SW を OFF に設定した場合の動作で固定されます。

表 1：DIP-SW と動作モードの関係

| | OFF | ON | 備考 |
|-----|------------|--------------|----------|
| SW1 | USB1.1 モード | USB2.0 モード | 下記※印参照 |
| SW2 | | | (予約) |
| SW3 | | | (予約) |
| SW4 | 通常運用モード | ファームウェア更新モード | 現行版では非対応 |

※ USB2.0 モードに設定した場合でも、high speed に対応していない USB ポートに接続されると、full speed での転送になります（USB 規格準拠の動作）。ホスト PC 側では、FW_VERSION（☛ 表 7）の MSB によって、実際に稼動している速度を確認することができます。

※ high speed 動作時と full speed 動作時では、USB 転送の際の単位データサイズが異なりますのでご注意ください（☛ 4.2 節、☛ 表 5）。USB2.0 を想定していない PC アプリを使用する場合は、「USB1.1 モード」に設定して下さい。

2．SPI/USB 変換ユニット本体の LED 表示

変換ユニットの、LED（緑）と LED（赤）の表示パターンは、下表のようになります。

表 2：LED 表示パターン

| 状態 | LED（緑） | LED（赤） | 備考 |
|---------------|--------|--------|---|
| ユニット電源 OFF | 消灯 | 消灯 | |
| FPGA 起動エラー | 消灯 | 点灯 | この状態が発生した場合、弊社にご連絡下さい |
| SPI 電源 OFF 検出 | 点滅 | | (500 ミリ秒周期) |
| SPI 電源 ON 検出 | 点灯 | | |
| USB データ転送不能 | | 点滅 | ・ホスト PC にドライバがインストールされていない、または、ホスト PC による USB リセット状態 (500 ミリ秒周期) |
| | | | ・ホスト PC による USB サスペンド状態 (点灯 125 ミリ秒／消灯 875 ミリ秒の繰り返し) |
| | | | ・上記以外の転送不能状態 (250 ミリ秒周期) |
| USB データ転送可能 | | 消灯 | |
| USB データ転送中 | | 点灯 | (送受信処理期間 + α ：下記※印参照) |

※ USB 転送処理時の LED（赤）の点灯期間は、短い USB 通信の場合でも目視できるように、実際の処理時間よりも 125 ミリ秒長く取っています。

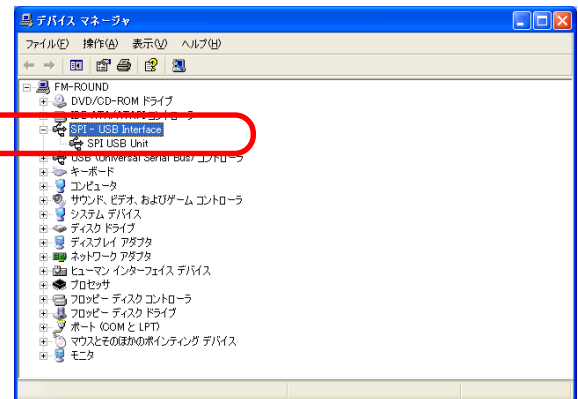
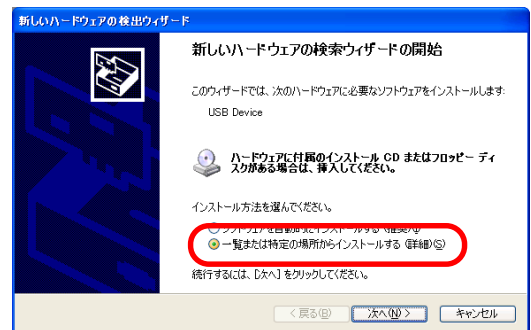
※ ファームウェア更新処理中（☛ 表 1）は、「LED（緑）＝消灯、LED（赤）＝高速点滅」の状態になります。

3．ホストP Cとの接続

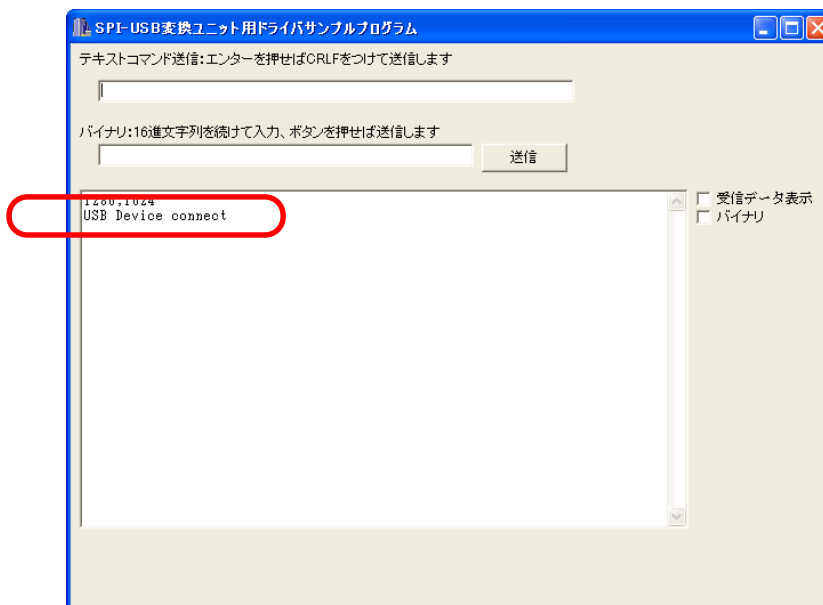
変換ユニットを初めてU S Bに接続した時には、ホストP Cの「新しいハードウェアの検出ウィザード」が起動します（自動的に起動しない場合は、ハードウェア追加ウィザードを手動で起動して下さい）。

ウィザードからU S Bドライバを要求されたら、**CD-ROM**ドライブに付属の**CD**を挿入し、¥driver フォルダを指定して下さい。

ウィザードが終了したら、U S Bドライバが正常にインストールされたことをデバイスマネージャで確認して下さい（ホストP Cと正常に接続すると、変換ユニットの**LED**（赤）が「点滅」から「消灯」に変化します：表2）。



次に、ご使用ホストP C環境で**DLL**が正常に動作できることを確認するために、¥dll¥Sample.exe を起動して下さい。「USB Device connect」が表示されれば、変換ユニットとのU S B通信が可能です。



※ Sample.exeは、ご使用環境でのU S Bドライバの動作対応確認用です。本仕様で定義している**USB**通信パケット処理機能を実装したものではありませんが、手動入力によって、変換ユニットに仕様定義のコマンドパケットを送信することも可能です。「受信データ表示」「バイナリ」の両方にチェックを入れ、バイナリエディットボックスに、例えば、“52000001000001”（000001h番地からの1バイトリード）を入力して「送信」ボタンをクリックすると、変換ユニットからファームウェアバージョンが返信され、その内容が画面上に表示されます。

4．P Cアプリケーションの開発

変換ユニットは、ホストP Cに対してU S Bデバイスとして動作します。

4．1 DLL の利用

P CアプリからU S Bドライバへのアクセスは、**DLL** を介して行います。

F2Usb.h をインクルードし、F2Usb.dll を動的リンクして、**DLL** の各 **API** を呼び出して下さい。

※各**API**の詳細については、F2Usb.hのコメントを参照して下さい。

DLL に対するアプリケーションの基本的な処理は、以下のようになります。

- ① アプリケーション開始時に Usb_Create によって監視スレッドを作成して下さい。
- ② USB イベント処理用関数 foo を作成し、foo を Usb_SetEventRoutine によって登録して下さい。
変換ユニットの接続／切断イベント、変換ユニットからの受信イベント発生時には、登録済みのイベント処理用関数 foo がコールバックされます。接続／切断イベントの場合は、対応する処理を foo 内に実装して下さい。受信イベントの場合は foo 内で Usb_Read をコールして受信データを取得し、適宜データを処理して下さい。
- ③ 変換ユニットへ送信する場合は Usb_Write または Usb_WriteText を使用して下さい。
- ④ アプリケーション終了時には Usb_Delete によって監視スレッドを解放して下さい。

4．2 U S Bインタフェース仕様

変換ユニットの **USB** 通信基本仕様を下表に示します。

※変換ユニットの**DIP-SW**（☛ 表 1）によって、**USB2.0**モードに設定した場合でも、**high speed** に対応していない**U S B**ポートに接続されると、**full speed** での転送になります（**USB**規格準拠の動作）。
ホストP C側では、**FW_VERSION**（☛ 表 7）の**MSB**（0⇒full、1⇒high）によって、実際に稼動している速度を確認することができます。

表 3：変換ユニットのU S B通信基本仕様

| 転送方式 | バルク転送 | 12 [Mbit/s] (full speed) | 480 [Mbit/s] (high speed) |
|-----------|---------------------------|--------------------------|---------------------------|
| EP0 | コントロール転送用 | ペイロード：64 [Byte] | ペイロード：64 [Byte] |
| EP1 (OUT) | データ転送用 (ホストP C⇒変換ユニット) | ペイロード：64 [Byte] | ペイロード：512 [Byte] |
| EP2 (IN) | データ転送用 (変換ユニット⇒ホストP C) | ペイロード：64 [Byte] | ペイロード：512 [Byte] |

※ EP1、EP2 のペイロードサイズは **USB** 規格上の制約です。

※ 転送データのスループットは、ホストP C側のフレームスケジューリング、他のデバイスによる **USB** バスの利用状況、変換ユニット及びP C側のソフトウェア実装に依存します。

4.3 USB通信パッケージ

制御コマンド（EP1）、SPIデータ（EP1、EP2）のフォーマットを、以下のように定義します。

※各フィールドは全てバイナリ形式です。

表4：制御コマンドパッケージ

| フィールド | コマンドID | レジスタアドレス | データ長（L） |
|-------|--------------------------------|-----------------|-----------------|
| サイズ | 1 Byte | 3 Byte | 3 Byte |
| 説明 | 読み出し：52h（‘R’） 書き込み：57h（‘W’） | 000000h～FFFFFFh | 000000h～FFFFFFh |

※ USB バルク転送を採用しているため、チェックサム等は設けていません。

表5：データパッケージ

| フィールド | データID | SPIデータ |
|-------|----------|----------|
| サイズ | 1 Byte | 1～U Byte |
| 説明 | 44h（‘D’） | 上記（L）に依存 |

※ SPI データ長 U は、表3のペイロードサイズー1として定義します。具体的な値は、FW_VERSION（表7：000001h）のMSBに応じて、次のようになります。

MSB=0 ⇒ full speed 接続 U=63

MSB=1 ⇒ high speed 接続 U=511

「USB1.1 モード」（表1）で運用している場合は、U=63 固定になります。

※ データIDはリード／ライト共通です（送受信で異なるEPを用いるため、弁別不要）。

表6：書き込み完了通知パッケージ

| フィールド | 通知ID |
|-------|----------|
| サイズ | 1 Byte |
| 説明 | 46h（‘F’） |

受信パッケージエラーについて

変換ユニットでは、以下のいずれかの場合、パッケージエラーと見なし、その時点で受信済みの全てのバイト列を破棄します（バイナリスタイルのため、パッケージ同期の取り直しはUバイト単位でしか行えません）。

- 受信バイト列の先頭バイトが‘R’、‘W’、‘D’のいずれでもない（構文エラー）
- 先頭バイトが‘R’または‘W’の場合に、受信バイト列長が7未満（引数不足）
- 先頭バイトが‘D’の場合に、受信バイト列長が2未満（引数不足）
- 正常な‘W’パッケージ受信後、次の受信先頭バイトが‘D’以外（文脈エラー）
- ‘W’パッケージを受信していない状態で、‘D’パッケージを受信（文脈エラー）
- ‘W’パッケージによる書き込みデータ長指定 L に対して、 $\text{ceil}(L/U)-1$ 番目までのデータパッケージ長がU未満（データパッケージ不正分割：詳細については4.5(2)と4.5(3)を参照して下さい）

4. 4 レジスタマップ

変換ユニットは、ゲーム機に対してはSPIスレーブとして動作します。

SPIアクセス用制御レジスタ群は、下表のようにマッピングされています。

表7：変換ユニットの制御用レジスタ

| アドレス | レジスタ名 | SPI 属性 | USB 属性 | 概要 | (FPGA) |
|-------------------------|--------------|-----------|-----------|---------------------------|-------------------------|
| 000000h | FPGA_VERSION | R | R | 変換ユニットのFPGAバージョン情報 | 400000h |
| 000001h | FW_VERSION | R | R | 変換ユニットのF/Wバージョン情報 | 400001h |
| 000002h | POWER_DET | R | R | ゲーム機電源（POWER）検出状態 | 400002h |
| 000003h | | | | （予約：固定ゼロ） | 400003h |
| 000004h | COUNT_R0 | R | R | FIFO_R 格納済みデータ数（bit07:00） | 400004h |
| 000005h | COUNT_R1 | R | R | FIFO_R 格納済みデータ数（bit15:08） | 400005h |
| 000006h | COUNT_R2 | R | R | FIFO_R 格納済みデータ数（bit23:16） | 400006h |
| 000007h | | | | （予約：固定ゼロ） | 400007h |
| 000008h | COUNT_W0 | R | R | FIFO_W 格納済みデータ数（bit07:00） | 400008h |
| 000009h | COUNT_W1 | R | R | FIFO_W 格納済みデータ数（bit15:08） | 400009h |
| 00000Ah | COUNT_W2 | R | R | FIFO_W 格納済みデータ数（bit23:16） | 40000Ah |
| 00000Bh | | | | （予約：固定ゼロ） | 40000Bh |
| 00000Ch | COUNT_D0 | R | R | FIFO_D 格納済みデータ数（bit07:00） | 40000Ch |
| 00000Dh | COUNT_D1 | R | R | FIFO_D 格納済みデータ数（bit15:08） | 40000Dh |
| 00000Eh | COUNT_D2 | R | R | FIFO_D 格納済みデータ数（bit23:16） | 40000Eh |
| 00000Fh | | | | （予約：固定ゼロ） | 40000Fh |
| 000010h 00001Fh | | | | （予約：固定ゼロ） | 400010h 40001Fh |
| 000020h 000027h | GENERAL_RD | R | W | SPI 汎用受信用（USB 汎用送信用） | 400020h 400027h |
| 000028h 00003Fh | | | | （予約：固定ゼロ） | 400028h 40003Fh |
| 000040h 000047h | GENERAL_WR | W | R | SPI 汎用送信用（USB 汎用受信用） | 400040h 400047h |
| 000048h 000EFFh | | | | （予約：固定ゼロ） | 400048h 400EFFh |
| 000F00h 000FFFh | | | | （予約：変換ユニットFPGA内部制御） | 400F00h 400FFFh |

| | | | | | |
|-------------------------|--------|-----|---|------------------------|-------------------------|
| 001000h 001FFFh | FIFO_R | R | W | SPI リード用（USB ライト用）FIFO | 401000h 401FFFh |
| 002000h 002FFFh | FIFO_W | W | R | SPI ライト用（USB リード用）FIFO | 402000h 402FFFh |
| 003000h | RESET | (W) | W | FIFO 強制リセット | 400F07h |
| 003001h 007FFFh | | | | （予約：固定ゼロ） | |
| 008000h 008FFFh | FIFO_D | W | R | 画像データ用 FIFO | 403000h 403FFFh |
| 009000h FFFFFFh | | | | （アクセス禁止領域） | |

※ 「FPGA」欄は、変換ユニット内部制御用情報（＝弊社内部仕様）です。基本的に S P I アドレスに 400000h を加算した値になっていますが、FPGA メモリサイズ 16kByte に収めるために、変則的な割り当て箇所（色付き枠）が部分的に存在します（ゲーム機や P C アプリで意識する必要はありません）。

※ 「予約」及び「アクセス禁止領域」へのアクセスは、リード時 00h 固定、ライト無視になります（原則的に、これらの領域へのアクセスは避けて下さい）。

※ FW_VERSION（000001h）の MSB によって、実際に移動している USB 速度（0⇒full、1⇒high）を確認することができます。

FIFO について

- RESET が 01h のとき、各FIFOが有効になります。RESET が 00h 状態の場合、FIFOへのリード／ライトは無効です（変換ユニット起動直後はFIFOは有効（RESET=01h）になっています）。
- FIFOをリセットする場合は、RESETに01h⇒00h⇒01hを書き込んで下さい。COUNT_R / COUNT_W / COUNT_D が全てゼロになります（各FIFOのデータ自体はクリアされません）。
- FIFOに割り当てられているメモリ空間の、どのアドレスにアクセスしてもFIFOの送受信を行うことができます（バーストモード対応）。
- 各FIFOの容量は4096Byteです。
- FIFOカウンタが 000000h（データ無し）のときにリードアクセスを行った場合に読み出される値は不定です（FIFOカウンタは 000000h を維持）。
- FIFOカウンタが 001000h（データフル）の状態でのライトアクセスは無効です（FIFOカウンタは 001000h を維持）。

4.5 USB通信フロー

変換ユニットに対するPCアプリの基本的な処理は、以下のようになります。

- 変換ユニットの接続／切断状態に応じて、USB送受信操作の許可／禁止を制御
- USB読み出しコマンド（☛ 表4）の送信、及び、データパケット（☛ 表5）のUSB受信
- USB書き込みコマンド（☛ 表4）、データパケット（☛ 表5）のUSB送信、及び、書き込み完了通知パケット（☛ 表6）のUSB受信

以下では、可読性を考慮して、USBパケットの各フィールド毎に“:”を入れて表記しています（実際のパケットには“:”は存在しません）。また、便宜上、データフィールドをレジスタ名で表記している箇所があります。

(1) 接続確認処理

(1-a) 変換ユニットがUSB接続されていることを認識したら、52:000001:000001 (FW_VERSION リード要求) を発行します。変換ユニットは、44:FW_VERSION を応答します。FW_VERSION のMSBが0ならばfull speed接続なので、**U=63**とし、MSBが1ならばhigh speed接続なので、**U=511**としておきます。

(1-b) 変換ユニットがUSB接続されている期間中は、52:000002:000001 (POWER_DET リード要求) を周期的に発行して、ゲーム機の電源状態を確認します（変換ユニットは、44:POWER_DET を応答します）。

(1-c) POWER_DET=01hになるまで(1-b)を繰り返します。

※ ゲーム機の電源投入後、実際にPOWER_DET=01hになるまでに数十ミリ秒を要します（ゲーム機側でも同様に、POWER_DET=01hになるまで繰り返しSPIリードして下さい）。

(1-d) RESETに01h⇒00h⇒01hをライトすることでFIFOをリセットすることができます。

※ FIFOをリセットする場合は、

57:003000:000001 と 44:01 の連続発行＋書き込み完了通知 46h の受信待ち

57:003000:000001 と 44:00 の連続発行＋書き込み完了通知 46h の受信待ち

57:003000:000001 と 44:01 の連続発行＋書き込み完了通知 46h の受信待ち

という3往復の通信が必要です。

※ これ以降も、常に(1-b)を実施し、データ転送の途中でPOWER_DETが00hに変化した場合は、処理を中止して(1-c)に戻して下さい。

実際のUSB転送速度とペイロードサイズについて

- ご利用環境に於いて、ホストPCのUSBポートがhigh speed対応であることが確実に保証されており、かつ、ファームウェアバージョンが02h以降の変換ユニットをご使用の場合は、上記(1-a)の手順は省略可能です（常に**U=511**と見なすことができます）。
- ファームウェアバージョンが01hの変換ユニットをご使用の場合は、常にfull speed接続になるため、**U=63**として上記(1-a)の手順を省略することができます。
- high speed非対応ポートが存在する可能性が有る場合、及び、バージョンの異なる変換ユニットが混在する可能性が有る場合は、上記(1-a)の手順を**必ず実施**して下さい。

(2) ホストPCからゲーム機へのデータ転送手順（600バイト転送の例）

※ この方向の場合は、**FIFO_R**に対するホストPCからのライト処理になります。

※ ここでは、**600**バイト分の転送を例示します。

(2-a) PCアプリは、52:000004:000003（COUNT_R リード要求）を発行し、COUNT_R 値が 4096-600=003A96h 以下である（＝空きが 600 バイト以上有る）ことを確認します。

※ FIFO_R の空き容量が不足している場合は、ゲーム機側からのリードによって FIFO_R が空くまで待って下さい（4. 5 (1-d) によって FIFO を強制リセットすることも可能ですが、FIFO_R だけでなく、FIFO_W、FIFO_D もリセットされることに注意して下さい）。

(2-b) PCアプリは、57:001000:000258（FIFO_R への 600 バイトデータ書き込み開始要求）を発行します。

(2-c) PCアプリは、データパケット（44:data×U,...：最後のデータパケットは 44:data×U 以下）を送信し、書き込み完了通知（46h）が受信されるまで待ちます。

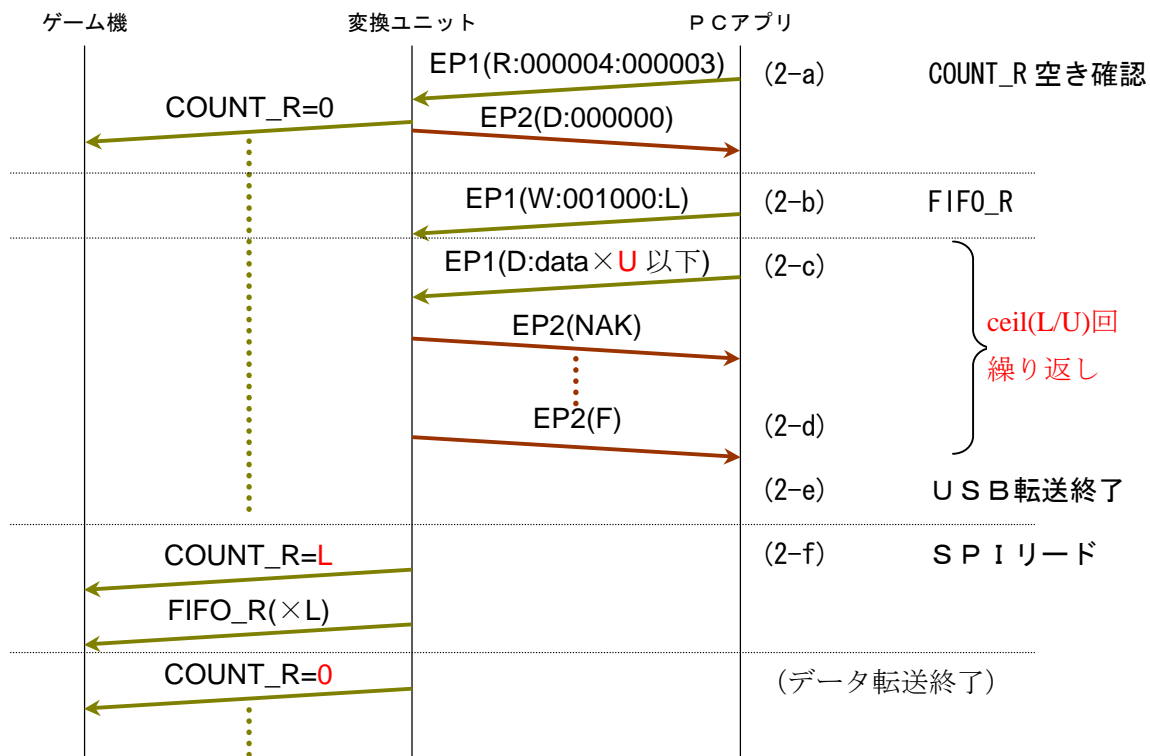
(2-d) 変換ユニットは、受信した U [Byte] 分のデータを FIFO_R に格納し、書き込み完了通知（46h）を送信します。

(2-e) 転送済み累積データバイト数が 600 になるまで、(2-c) (2-d) を繰り返します。転送済み累積データバイト数が 600 になったら、4. 5 (1-b) に戻ります（USB 転送終了）。

(2-f) ゲーム機は、COUNT_R 値>0 状態を検出した場合、COUNT_R 値分だけ FIFO_R からデータをバーストリードします。

※ この例のように、要求データ長 L が U [Byte] を超えている場合は、(2-c) (2-d) の処理を $\text{ceil}(L/U)$ 回繰り返すことになります。

※ 変換ユニットが各レジスタへのデータ格納処理を行っている状態、及び、アイドル状態に於いては、EP2 が NAK 応答（PCアプリ側の受信イベントが発生しない状態）になります。



※ FIFO 容量 (4kByte) を超えるデータを転送する場合、上記フロー全体を必要回数繰り返す必要があります。

(3) ゲーム機からホスト P C へのデータ転送手順（800 バイト転送の例）

※ この方向の場合は、**FIFO_W** に対するホスト P C からのリード処理になります。

※ ここでは、**800** バイト分の転送を例示します。

(3-a) ゲーム機は、S P I から 000008h、000009h、00000Ah をリードして、COUNT_W 値が 4096-800=000CE0h 以下である（＝空きが 800 バイト以上有る）ことを確認します。

※ FIFO_W の空き容量が不足している場合は、ホスト P C 側からのリードによって FIFO_W が空くまで待つ下さい（4.5 (1-d) によって FIFO を強制リセットすることも可能ですが、FIFO_W だけでなく、FIFO_R、FIFO_D もリセットされることに注意して下さい）。

(3-b) ゲーム機は、ホスト P C に転送したいデータ列を、FIFO_W (02000h) にライトします（S P I 転送終了）。

(3-c) P C アプリは、周期的に 52:000008:000003 を発行して、COUNT_W 値を確認します。
COUNT_W 値>0 状態を検出した場合（このときの COUNT_W 値を L と表記します）、52:002000:L（FIFO_W からの L バイトデータ読み出し開始要求）を発行します。

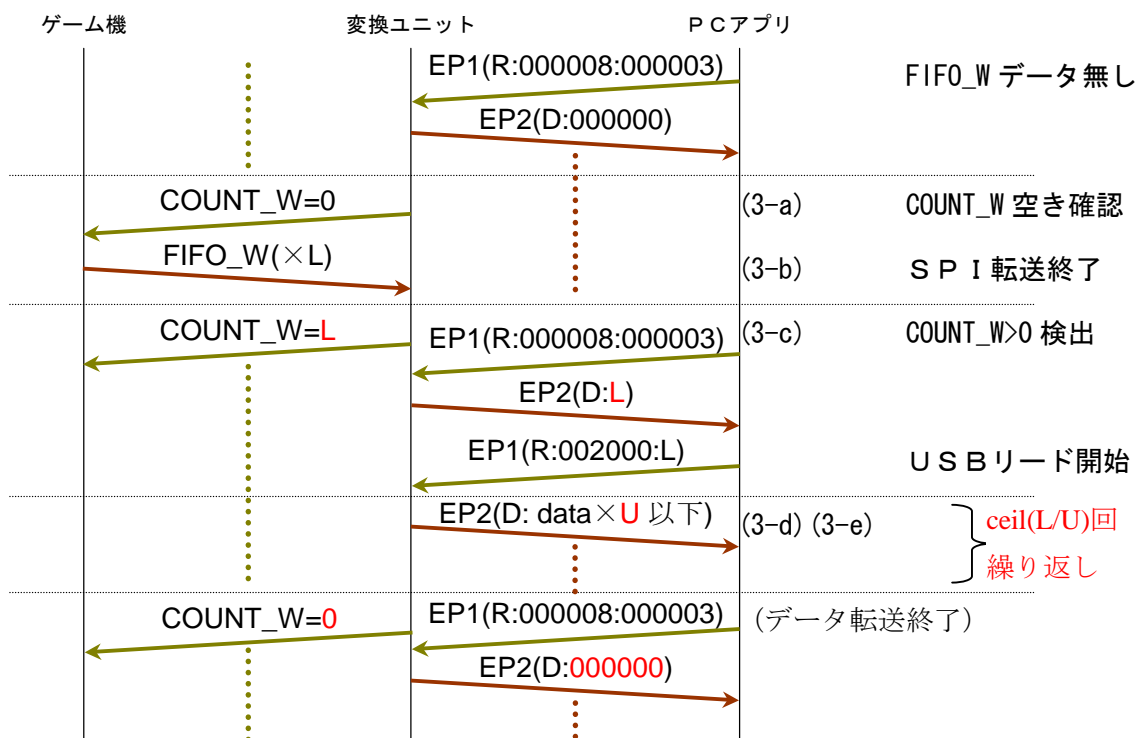
(3-d) 変換ユニットは、L バイト分の FIFO_W データを U [Byte] 毎に分割して、 $\text{ceil}(L/U)$ 個のデータパケット（44:data×U,... : 最後のデータパケットは 44:data×U 以下）を順次送信します。

(3-e) P C アプリは、L バイトのデータが揃うまでデータパケット受信を繰り返します。

※ この例のように、データ長 L が U [Byte] を超えている場合は、(3-d) (3-e) で $\text{ceil}(L/U)$ 回のデータパケット転送が発生することになります。

※ **FIFO_D と COUNT_D を利用する場合**の手順は (3) と同様です。

※ 変換ユニットが U S B 送信するための前処理を行っている状態、及び、アイドル状態に於いては、EP2 が NAK 応答（P C アプリ側の受信イベントが発生しない状態）になります。



※ FIFO 容量（4kByte）を超えるデータを転送する場合、上記 **フロー全体** を必要回数繰り返す必要があります。

4. 6 通信所要時間の評価

(1) full speed 接続：U=63 の場合

full speed 接続の場合、エンドポイントのペイロードサイズは最大 64 [Byte]に制限されます（☛ 表 3）。このうち 1 バイト分はデータパケット ID として使用するため（☛ 表 5）、L [Byte]のデータを USB 転送するには、 $\text{ceil}(L/63)$ 回のパケット転送が必要です。ここで、OS による USB フレームのスケジューリングが 1 [msec⁻¹]（理想値通り）であると仮定すると（実測値については、表 9 参照）、ホスト PC からゲーム機へのデータ転送のスループット $T_1(L)$ は、4. 5 (2) より、

$$\begin{aligned} (2-a) \quad (2-b) & \quad 3 \text{ [msec]} \\ (2-c) \quad (2-d) & \quad 3 \text{ [msec]} \times \text{ceil}(L/63) : \text{NAK は 1 回と仮定} \\ (2-f) & \quad S_1 \text{ [msec]} : \text{S P I 側の処理時間} \\ \therefore T_1(L) &= 3 + 3 \times \text{ceil}(L/63) + S_1 \text{ [msec]} \end{aligned}$$

で与えられます。同様に、ゲーム機からホスト PC へのデータ転送スループット $T_2(L)$ は、

$$T_2(L) = 3 + 2 \times \text{ceil}(L/63) + S_2 \text{ [msec]}$$

となります。以上のことから、 $L \leq 4096$ [Byte]の場合のスループット見込み値は、下表のようになります。

表 8：L ≤ 4096 の場合のスループット見込み値（full speed 接続時）

| L [Byte] | ceil(L/63) | $T_1(L)$ [msec] | $T_2(L)$ [msec] |
|----------|------------|-----------------|-----------------|
| 1 ~ 63 | 1 | $6 + S_1$ | $5 + S_2$ |
| 64 ~ 126 | 2 | $9 + S_1$ | $7 + S_2$ |
| | | | |
| 4096 | 66 | $201 + S_1$ | $135 + S_2$ |

また、 $L > 4096$ [Byte]の場合は、4. 5 の(2)や(3)のフロー全体を必要な回数（I と表記します）だけ繰り返すこととなります。例えば、24 ビット RGB-VGA データの場合、 $L=640 \times 480 \times 3=921600$ [Byte]、 $I=L/4096=225$ なので、USB 転送時間は下式のように、送信時約 4 5 秒、受信時約 3 0 秒となります。

$$I \times T_1(4096) = 45225 + (225 \times S_1) \text{ [msec]}, \quad I \times T_2(4096) = 30375 + (225 \times S_2) \text{ [msec]}$$

なお、弊社環境で実測したところ、下表のように、OS による USB フレームスケジューリング（IN-Token の周期）が理想値よりも遅い（下表では、理想値の 28%乃至 42%程度しか速度が出ていません）という結果が観測されました（このとき、変換ユニットの処理オーバーヘッドによる NAK 返信ケースは無し）。したがって、最終的なスループットは、OS に依存する要素が大きいと言えます。

表 9：USB フレームスケジューリング実測平均値（full speed 接続時）

| PC | | | OS | IN-Token 発行頻度 |
|-----|-----------|-----------------------------|-------------|---------------|
| 富士通 | FMVXD0202 | Celeron 2.80GHz / RAM 256MB | WinXP (SP2) | 平均 2.4msec 毎 |
| NEC | MJ26X/R-j | Celeron 2.66GHz / RAM 1GB | WinXP (SP2) | 平均 3.6msec 毎 |

(2) high speed 接続：U=511 の場合

high speed 接続の場合、エンドポイントのペイロードサイズは **512 [Byte]**固定になります（☛ 表 3）。このうち 1 バイト分は **full speed** 接続時と同様、データパケット ID として使用するため（☛ 表 5）、**L [Byte]** のデータは **ceil(L/511)** 個の U S B パケットに分割されます。原理的には、**high speed** 接続では **125 [μ sec]** 毎に 1 パケットの転送が可能ですが、各パケット毎の変換ユニット側の処理オーバーヘッド実測値が約 **500 [μ sec]**（2010/06/23）であるため、**ホスト P C 側の遅延が無いと仮定**した場合、**RGB-VGA** データの転送時間の見込み値は下式のようにになります（U S B 通信時間は約 1 秒程度）。

$$T'_2(4096) = 2 \times 0.125 + \text{ceil}(4096/511) \times 0.5 + S_2 = 4.75 + S_2 \quad [msec]$$

$$\therefore I \times T'_2(4096) = 1068.75 + (225 \times S_2) \quad [msec]$$

2010/07/22

実環境の場合には、ホスト P C 側の **IN-Token** 発行の遅れ、**FIFO** カウンタ値確認処理のオーバーヘッド等により、**900 [kByte]** 分のデータ転送処理時間は約 **2 秒** という結果になりました（ホスト P C は、表 9 の FMVXD0202）。つまり、**RGB-VGA** 2 面分のデータ転送時間は **4 秒程度** ということになります。

※ ゲーム機による撮像処理～データ変換処理時間は含まれていません。

また、同じ環境で、別の **USB** ポートに他の **USB** デバイスを追加接続していくと、約 **100 ミリ秒** 程度ずつデータ転送時間が増加していくことが観察されました。更に、**USB** メモリからのファイルコピー中に於いては、データ転送時間が 2 倍近くまで伸びることを確認しました。

ROUND 株 式 会 社 ラ ウ ン ド

＜本社＞

〒611-0011 京都府宇治市五ヶ庄芝東3番地9

■ T E L 0774-33-5282 (代)

■ F A X 0774-33-5297

■ 電子メール round@round.ne.jp

■ インターネットホームページ <http://www.round.ne.jp/>

＜東京営業所＞

〒162-0845 東京都新宿区市谷本村町3-19 千代田ビル4階

■ T E L 03-5225-9350

■ F A X 03-5225-9351

※電子メール、インターネットホームページは共通です。