USB2.0 を使用した、USB2-VME 制御モジュールの開発

佐藤節夫

高エネルギー加速器研究機構 中性子科学研究施設

概要

最近、パーソナルコンピュータに標準装備されてきている USB2.0 インターフェースを使用した、USB2-VME 制御モジュールの開発をしましたので、紹介します。USB2.0 インターフェースが付いていれば、小型なノートパソコンでもデータ収集できるため、応用範囲が広がります。

1　開発の経緯

高エネルギー加速器研究機構の中性子科学研究施設では、データ収集システムを VME 規格に沿って開発しています。一般に、VME モジュールを制御する CPU モジュールは、大量生産されるパーソナルコンピュータに比べて、高価で時代遅れなものになります。そのため、約２０年前から CPU モジュールを使用しない方法を開発してきました。当時からマッキントッシュに標準装備されていた SCSI インターフェースを使用して、VME モジュールを制御する SCSI-VME 制御モジュールを作製し、低価格で効率的な中性子実験用のデータ収集システムを提供してきています。図１に中性子測定システム（文献 2）の構成例を示します。SCSI-VME 制御モジュールは”SCSI-VME module”と書いてある、ダブルハイト、１枚幅の VME モジュールで、CPU モジュールの代わりをしています。パーソナルコンピュータの単なる入出力機器として VME 測定器を制御できるようになります。パーソナルコンピュータとしてはマッキントッシュ（MacOS）か Windows98/Me が使用できます。

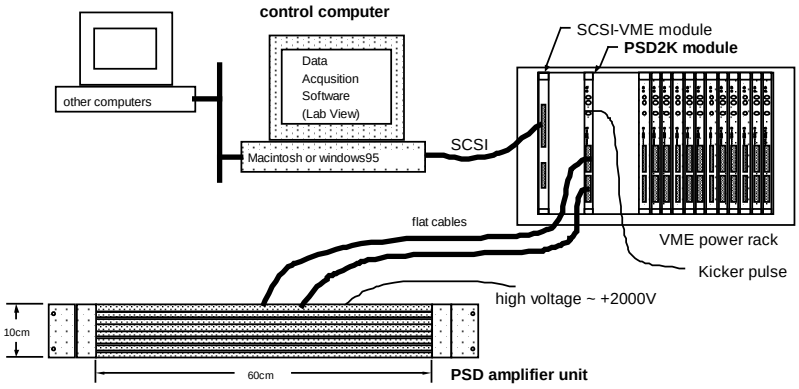

図 1.』 中性子測定システムの構成例（PSD2K システム）

しかし、数年前から SCSI インターフェースが標準装備されなくなってきたため、置き換えの開発をしてきています。これまでにも、USB1.1-SCSI 変換アダプタや、Ethernet10B-SCSI 変換アダプタを開発していますが、転送速度が遅く、データ容量が小さい実験に限られてしまっていました。そこで、最近の USB2.0 の登場で、SCSI インターフェースの転送速度に匹敵してきたために、十分に置き換えが可能になってきていましたが、高速になった分だけ使用方法が高度になり、一般のユーザには手におえない領域でした。しかし、サイプレス社が非常に使いやすい製品を出したため、開発が可能になりました。

データ収集システムのメインアプリケーションとしては、当初からナショナルインスツルメンツ社の LabVIEW を使用していました。グラフィカルなプログラムの開発が可能で、マッキントッシュでも Windows でも同じ環境で使用できます。そして、データの入出力部分を入れ替えるだけで、SCSI、USB1.1、Ethernet を簡単に切り替えることができます。今回の開発で、USB2.0 も選べることになり、選択の幅が広がりました。

2　USB2-VME 制御モジュール

図２に、今回開発した USB2 -VME 制御モジュールを示します。図１の SCSI-VME 制御モジュールに置き換えられます。

母体となる基板は、今までに開発している測定器用を使用しています。手前に見えている手作りの増設基板に USB2.0 のコントローラが付いています。見えにくいですが、USB コネクタも付いていて、コンピュータに接続します。左上に見えるフィールドプログラムブルゲートアレイ（FPGA）で VME 側の制御と、USB2.0 コントローラの制御を行っています。

図 2.』 USB2-VME 制御モジュール

2.1　USB2.0 の開発環境

USB2.0 のコントローラにはサイプレス社の CY7C6813-100AC（FX2）を採用しました。開発環境や、豊富なサンプルプログラムなどが無償でダウンロードでき、IC だけを買えば本格的な開発ができてしまいます。テストプラグラムのロードも USB コネクタを通してできますので、優れた設計の IC と言えます。開発後は、EEPROM からプログラムをロードできるように変更できます。USB からプログラムがダウンロードできる、インテル社のマイコン 8051 の C 言語開発モニタとしても使用できます。（文献 1）

2.2　FX2 のダウンロードと使い方

http://www.cypress.com/design/progprods/usb/devtools/an2131-dk001.html から EZ-USB コントロールパネルをダウンロードします。解凍して setup.exe を実行するだけです。ここでは標準のフォルダに展開した場合で説明していきます。細かい仕様については C:¥Cypress¥USB¥Doc¥FX2 内の FX2 TechRefManual.pdf ファイル等を参照してください。

C:¥Cypress¥USB¥Examples¥EzUsb¥bulktest¥target 内のプログラムを例にします。USB の標準的な使い方として、fw.c、descr.a51、dscr.h は常に同じにしているようです。fw.c には main()が含まれていて、標準的な処理を行います。descr.a51 と dscr.h はデバイスディスクリプタ等を定義しています。ここを変えれば好きなベンダ ID やプロダクト ID、エンドポイント数の変更ができます。

実際の動作の指定は、その他のサブルーチンプログラムで行います。この場合は periph.c がそれに相当し、TD_Init()、TD_Poll()等、決められた名前のサブルーチンを準備し、その中の動作を書き換えます。決められた名前の割り込みテーブルも準備し、必要に応じて中身を書き換えます。

EZ-USB マイコンは、ジャンプテーブルで USB の割り込みに入ってから、割り込みテーブルで飛び先を指定できます。C:¥Cypress¥USB¥Target¥Lib¥Ezusb 内の USBJumpTb.OBJ ファイルに登録してあります。そして、periph.c 内に同じ名前で処理関数を定義してあります。interrupt 0 の付いた不思議な関数が沢山ありますので、すぐに見つかると思います。

この他に、同じフォルダにある Ezusb.lib がプロジェクトに登録されています。Ezusb.lib の関数の使い方は C:¥Cypress¥USB¥Doc¥EZ-USB General 内の Anchor Library.pdf ファイルに書いてありましたが、若干フォルダ名とかが違っているので、古いものと思われます。このライブラリが持つ関数の説明もあり、サンプルプログラムを読むのに参考になります。関数の頭に EZUSB_が付いているものです。

2.3　FPGA からの FX2 の制御と動作原理

FX2 の使用方法としては、スレーブ FIFO とジェネラルプログラマブルインターフェース（GPIF）があります。ここでは、VME 規格に変換するために、どうしても FPGA を使用しなければならないので、スレーブ FIFO を使用しました。一般には、GPIF を使うほうが外部回路が減って有利ですが、使い方が非常に難しいことがわかりましたので、無理をしないことにしました。スレーブ FIFO は非常にわかりやすく、ほぼサンプルプログラムがそのまま使用できます。そのため、FX2 に深入りしなくともバグの少ないプログラムが書けてしまいます。

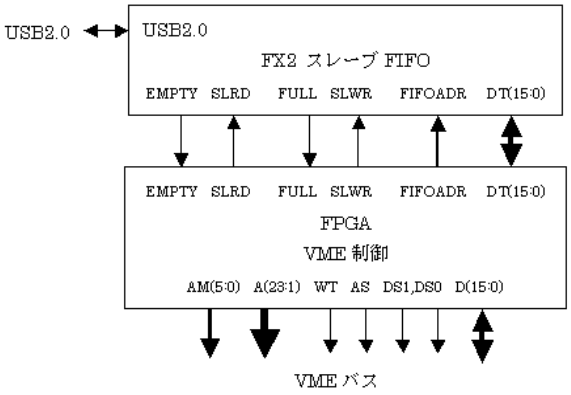

図 3.』 FPGA で FX2 を制御するブロック図

図３に FPGA で FX2 を制御するブロック図を示します。FX2 が USB2.0 の処理を全て行います。そして同期クロックも発生させ、FPGA の基本クロックとする、同期モードで使用しました。クロックを別々に持たせる、非同期モードもありますが、同期モードの方が高速動作します。

FPGA から見ると、FX2 は FIFO メモリに見えます。４つのポートを使用できますが、ここでは２ポートだけ使用し、入力と出力の FIFO メモリに設定しています。FPGA 側から見て入力に指定した FIFO のデータを調べ、あれば読み出します。出力に指定した FIFO が空いていたらデータを書き込みます。FIFO は 2 ビットのアドレスピン（FIFOADR）で切り替えて使用します。ここでは１ピンを固定し、もう１ピンだけで切り替えています。

FIFO の状態を示す信号は２ピンだけで、EMPTY（空）と FULL（満杯）です。入力の FIFO が EMPTY でなければ、ホストから送られたデータがあるので読み出します。出力の FIFO が FULL でなければ、FIFO が満杯になるまでデータを書き込みます。有効なデータ転送が始まらないうちは、読み出したデータは捨て、書き込むデータは 0 など、無意味なものとします。

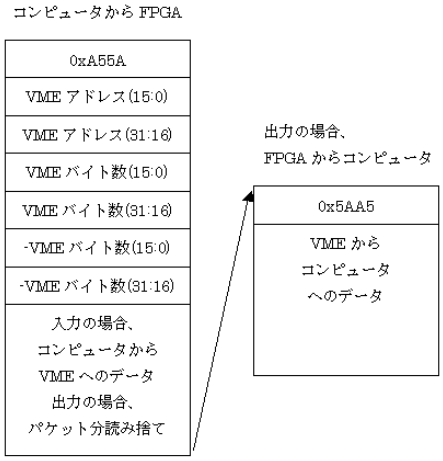

図 4.』 データ形式

図４にデータ形式を示します。このモジュール用に私的に決めたもので、データ長は 16 ビットです。入力の FIFO から 0xA55A が送られてきたところからデータ転送を開始します。初めの 14 バイトで転送開始の VME アドレスと、転送数を読み出し、VME 側に設定します。ホストから VME モジュールへのデータ転送であれば、引き続きデータを読み出し、転送数が終わるまで送り続けます。VME モジュールからホストへの転送であれば、データがなくなるまで（１パケット分）読み続け、捨てて、出力の FIFO に切り替えて、0x5AA5 を書き込んでデータの初めを知らせ、送りたいデータがなくなるまで書き続けます。

FPGA としては、XILINX 社の Virtex シリーズの XCV50-PQ240-5 を使用しました。本来は VME 測定モジュールに使用しているものですが、VME コントローラにも使用してしまいました。VME バスに直接アクセスするには、規格上容量不足ですが、小規模なシステムや、テストに限定して使用します。十分な動作が確認できた時点で、バッファなどを取り付け、規格を満たしたいと思っています。

2.4　FPGA のプログラム

開発環境としましては、XILINX 社の ISE 5.2 を使用しました。使用言語として、VHDL か Vlog が選べますが、VHDL で設計しました。それまではグラフィカルな CAD を使用していましたが、XILINX 社でサポートしなくなり、代わりに出してきたシステムです。グラフィカルな環境に慣れてしまっていましたので、時代に逆行するものだと思っていましたが、部品の配置や、配線の煩雑さもいらず、意外に使いやすいものでした。しかし、動作を想像しながらのテキスト文での設計は、慣れるまでに時間がかかりました。一行で 32 ビットの加算ができてしまうのには驚きました。まだまだ、今まで図面で描いていた回路の置き換えに苦慮しています。

3　パーソナルコンピュータのドライバ

3.1　WindowsXP

サイプレス社の開発ソフトが Windows 用に書かれていて、ソースも付いているので、参考にしました。ビジュアル C++で DLL ファイルを作るだけで LabVIEW のライブラリ関数に関連付けができました。

3.2　MacOSX

MacOSX のインストールセットに標準に付いている開発ソフトのプロジェクトビルダーに、サイプレス社 USB コントローラのサンプルプログラムが含まれていましたので、参考にしました。付属の C で framework ファイルを作るだけで LabVIEW のライブラリ関数に関連付けができました。ただし、MacOSX 用の LabVIEW7.0 は英語版しかなく、MacOSX10.2.8 では、英語版でインストールしないと関連付けができませんでした。今後、どちらかのバージョンが上がれば日本語版でも使える可能性があると思います。

4　動作確認・まとめ

動作確認としては、16M バイトのメモリの読み書きで行っています。16M バイトのモジュールを使用し、総合的なテストをしました。実測で、3～4 秒で書き込み、同様に 3～4 秒で読み出しが実行されました。モジュールの機能上、8M バイト／秒が最高速度ですが、シンクロスコープでの測定では 6M バイト／秒程度でした。トータルでは、さらにホスト側のスケージューリングが入るので、これよりも遅くなります。

データは 4 バイト長整数とし、0 から 1 づつ増やしたデータを 16M バイト書き込みました。そして同じ所から 16M バイト読み出し、違いがないかを確かめました。今まで、5000 回（8 時間程度）繰り返してもエラーがありませんでした。

現在は実験に使用し始めています。6M バイト／秒程度の転送速度が得られています。ほとんどの場合、問題なく使用できていますが、ハンドシェーク動作で、極端に転送速度が変化する装置ではうまく動作しないことがわかりました。内蔵の FIFO メモリが、これを吸収するはずですが、変化が大きい場合は不具合が生じます。自前で FIFO を持つなどの対策を考えています。

参考文献

[1]　佐藤節夫, “EZ-USB 使いこなしワンポイント”,CQ 出版社、トランジスタ技術、2003 年 3 月, p161~162

[2]　佐藤節夫, et al, “中性子散乱実験用位置敏感検出器、PSD2K システムの開発”, KEK Report, 2001-9 M/D

[3]　桑野雅彦, “ワンチップマイコン内臓 USB コントローラ AN2131SC を使った USB ターゲット機器の開発”, CQ 出版社, TECHI Vol.8』 USB ハード＆ソフト開発のすべて, 2000 年 5 月 15 日』 第４版