ＰＩＣ２４ＦＪ６４ＧＢ００２によるＵＳＢメモリの読み書き

小山工業高等専門学校　電子制御工学科　５年　本澤　上
指導教官　金野　茂男（http://www.oyama-ct.ac.jp/D/kinnoken/）

１、はじめに

一昔前は、ＰＩＣとパソコンの通信にＲＳ－２３２Ｃ等が利用されていた。しかし、現在ではそのためのポートがＰＣに用意されていないということも少なくない。そしてその代わりに、必ずパソコンに付いているのがＵＳＢポートである。ＰＩＣ１８Ｆシリーズではパソコンと通信するＵＳＢで接続するための機能が実装され、それに関する書籍も多数出版されている。だが、この場合のＰＩＣは、制御される側（デバイス）として動作するのみで、制御する側（ホスト）としての動作は難しいという状況であった。

このような中、登場したのがＰＩＣ２４ＦシリーズのＵＳＢホスト機能である。この機能は、正真正銘のＵＳＢホストとしてＵＳＢデバイスを制御するための機能であり、ＰＩＣの可能性を大きく拡げる画期的なものである。例えば、ＰＩＣを使用して製作した測定機器にＵＳＢメモリを接続し、測定データをＵＳＢメモリに保存しておけば、パソコンでの統計処理や、データの持ち運びに都合がよい。イメージとしては、下の図に示すようなものを考えている。

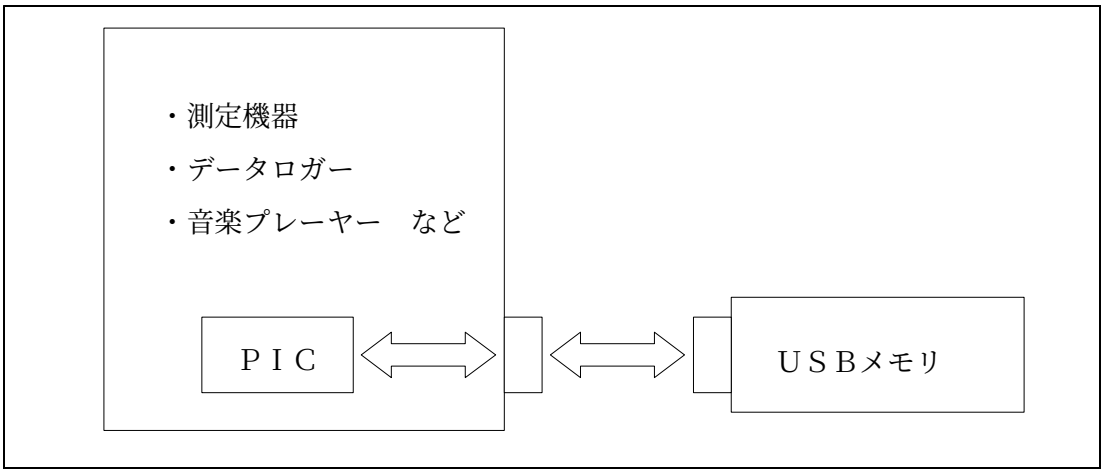

図１、活用のイメージ

２、開発環境

今回の研究では、ＰＩＣ２４Ｆシリーズの中でも、ピン数が少なく扱いやすいＰＩＣ２４ＦＪ６４ＧＢ００２（以下００２）を使用してＵＳＢメモリの読み書きを行うことを目的とする。開発ソフトは「ＭＰＬＡＢ　ＩＤＥ」、コンパイラは「Ｃ３０」を使用し、書き込みにはＰＩＣライタ「ＩＣＤ２」を用いた。いずれもマイクロチップ社の純正品である。バージョンは最新版であれば問題ないと思われるが、どうしても上手くいかないという場合は後に記すバージョンに合わせてみてほしい。

これらのソフトやライタを使うには、ある程度の経験が必要であると著者は考えている。不慣れな人にもわかるようにこの論文を書くつもりであるが、初歩的なソフトの使い方などについては、各自練習を重ねてほしい。

なお、読み書きをするＵＳＢメモリの容量は４ＧＢ（この程度あれば十分と思われる）とし、汎用性を確かめるために３社の製品を試した。

・デバイス：　　ＰＩＣ２４ＦＪ６４ＧＢ００２
・ＵＳＢメモリ：　　ＢＵＦＦＡＬＯ　ＲＵＦ－Ｃ４ＧＳ－ＢＬ／Ｕ２
　　ＥＬＥＣＯＭ　ＭＦ－ＬＳＵ２０４ＧＷＨ
　　ＳＯＮＹ　ＵＳＭ４ＧＬ－Ｗ
・ＰＩＣライタ：　　マイクロチップ社　ＩＣＤ２　（ICD2 Version - 08.43.02.02）
・開発ソフト：　　ＭＰＬＡＢ　ＩＤＥ　８．４６
　　Ｃ３０コンパイラ　３．２３
・ＰＣ：　　マイクロソフト　ＷｉｎｄｏｗｓＸＰ　Ｐｒｏ　ＳＰ３

３、ハードウェア

ＰＩＣでＵＳＢメモリにアクセスするにあたり、周辺回路が当然必要になってくる。００２のデータシートなどを参考に調べてみると、意外にも回路は簡単なものでよいことが分かった。そのため、ブレッドボード上でシステムを開発することにした。

図２、図３に、回路図と実際のブレッドボード上の回路を示す。ＵＳＢ周りの回路は、基本的に００２のデータシートに記載されている推奨回路と同じである。注意しなければならないのは、ＵＳＢコネクタのピン番号である。図４は、ＵＳＢメモリを接続した状態のＵＳＢコネクタの写真である。ピン番号は、ＵＳＢメモリに向かって右端が１番となっている。また、図４の左上部の回路は、ＩＣＤ２との接続のための回路である。ＩＣＳＰ（In-Circuit Serial Programming：回路にライタ接続用の端子を用意しておくことで、回路からＰＩＣを取り外さずにプログラムを書き込む方法）を使用しない場合は必要ない。ＩＣＳＰを行う場合は、ＩＣＤ２と回路とを繋ぐシリアルケーブルを図５、図６の様に加工しておくことをお勧めする。著者は、ケーブルにメスピンソケットを取り付けたものを使用している。参考にしてほしい。

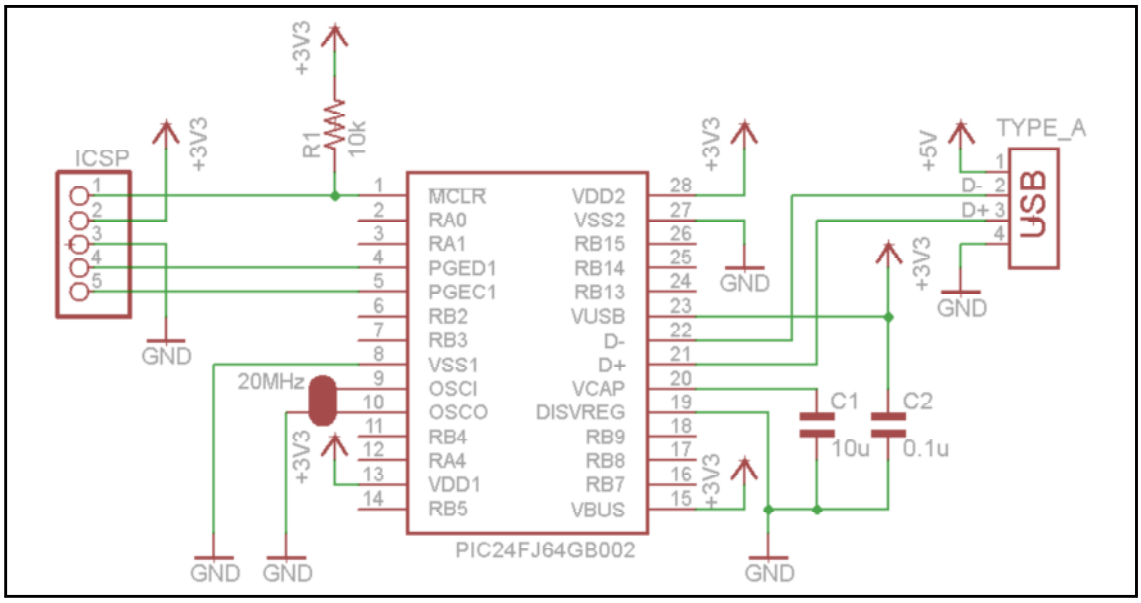

図２、システムの回路

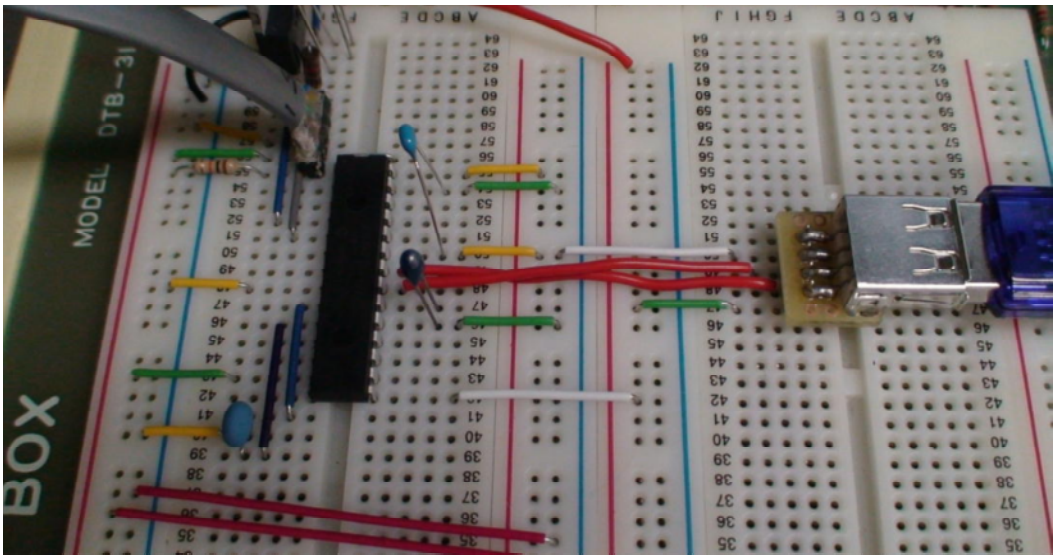

図３、実際の回路

図４、ＵＳＢメモリとコネクタ

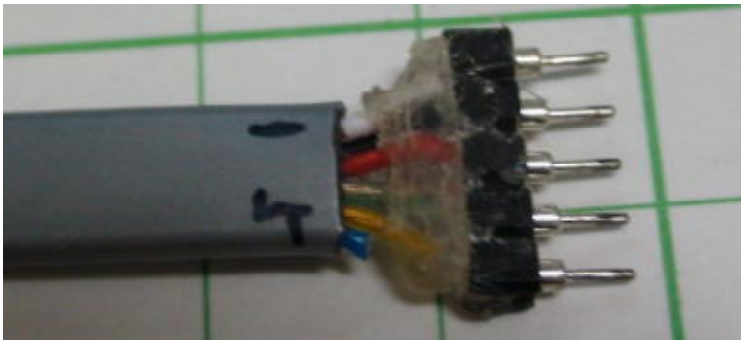

図５、ＩＣＳＰケーブルの加工部分の写真（ＰＩＣ側）

図６、末端を加工したＩＣＳＰケーブルの全体写真

４、ソフトウェア

ＰＣでもＰＩＣでも、ＵＳＢメモリを読み書きするためには、まずＵＳＢメモリとの相互の通信が出来なければならない。また、ファイルシステムを操作するプログラムも必要である。これらのプログラムを自力で作成することは、相当の実力と知識を持った人でなければまず無理である。そこで、マイクロチップ社のホームページから無料でダウンロードできるライブラリ"Microchip Application Libraries"を使用してＵＳＢ関連の一連の動作を行ってもらう。しかし、ライブラリを使うと言っても、インクルード処理やパスの設定などのやや複雑な作業があるため、慣れていない人はコンパイルを通すだけでも一苦労である。

そのため、先ほど述べた"Microchip Application Libraries"に含まれているサンプルを改変してプログラムを構成する。使用するのは"USB Host - Mass Strage - Simple Demo"である。これを著者が独自に００２用に書き換えたものをフォルダ"Mass_Strage"に入れてこの論文に添付する。各自、本研究室のホームページからダウンロードして頂きたい。以下、これをプロジェクトフォルダと呼ぶ。

４－１、プロジェクトの解説

プロジェクトフォルダには、次のファイル、フォルダが含まれている。

・ＦＳフォルダ：ＦＡＴファイルシステム関連のヘッダ、ソースが入っている。
・ＵＳＢフォルダ：ＵＳＢ関連のヘッダ、ソースが入っている。
・Ｃｏｍｍｏｎフォルダ：上記以外のヘッダ、ソースが入っている。
・Ｏｂｊｅｃｔｓフォルダ：コンパイラが出力するオブジェクトファイルのフォルダ
・ｍａｉｎ．ｃ：メインのプログラムソース
・Ｔｈｕｍｂ＿Ｄｒｉｖｅ．ｍｃｐ：ＭＰＬＡＢ用のプロジェクトファイル

また、コンパイル後に生成されるものとして、ｃｏｆ、ｈｅｘ、ｍａｐ、ｍｃｓというファイルがある。この中で最も重要なのはｈｅｘというファイルで、ＰＩＣに書き込まれるのがこのファイルである。

サンプルの改変点について述べる。まず、本来はライブラリに収められているヘッダファイルなどを、プロジェクトフォルダ内のＣｏｍｍｏｎ，ＦＳ，ＵＳＢというフォルダへ移した。これにより、プロジェクトフォルダを移動した際のパスの設定の手間を省くことが出来る。また、ｍａｉｎ．ｃ内の、他種デバイスのために書かれているコードは全て削除したが、デバッグ用に用意されている部分は、もしもの時のために残しておいた。ｍａｉｎ．ｃのコンフィグ設定は、ＰＩＣ２４ＦＪ６４ＧＢ００４用のものをほぼそのまま使用している。ただ、ＣＯＮＦＩＧ２のＰＬＬＤＩＶ＿ＰＬＬｘについては、ＰＩＣに供給しているクロックの周波数に応じてｘを変更する必要がある。計算式は

$$x = \frac{\text{クロック周波数}}{4\,\text{MHz}} \qquad (1)$$

である。今回の回路では２０ＭＨｚを使用したため、設定はＰＬＬＤＩＶ＿ＰＬＬ５となる。上の計算式からわかるように、クロック周波数は４の倍数で無ければならない。

４－２、開発について

この論文の大きな目的は、ＵＳＢメモリを使用したＰＩＣのアプリケーションの開発を手助けすることにある。しかし、このサンプルだけでは開発できるだけの知識は到底身に付かないであろう。そのため、マイクロチップ社のライブラリやそのマニュアルに、一度は目を通しておくことを強くお勧めする。開発において、関数の詳しい仕様や関数の内部を知っておくことはとても重要である。

５、システム構築の手順

以上で述べたソフト、ハードを組み合わせれば、ＵＳＢメモリを制御することが出来るが、コンパイル手順などがよく分からない人のために、細かい手順を説明する。とくに説明が要らないという人は読み飛ばして構わない。

①図２などを参考に、回路を作成する。このとき、まだＵＳＢメモリは接続しない方がよいであろう。
②ＭＰＬＡＢを起動し、ツールバーの「Open Project」からプロジェクトファイルである Ｔｈｕｍｂ＿Ｄｒｉｖｅ．ｍｃｐを開く。
③「Configure」→「Select Device」を開き、ＰＩＣ２４ＦＪ６４ＧＢ００２を選択する。
④回路の電源をＯＮにして、「Programmer」からＩＣＤ２を起動する。
⑤「Project」→「Build All」を行った後、「Programmer」→「Program」としてプログラムを書き込む。
⑥書き込みが終了したら、ＩＣＤ２を回路から外し、一度電源を切ってからＵＳＢメモリを接続する。
⑦再度電源を入れ、ＵＳＢメモリのアクセスランプが点滅することを確認する。これにより、ＰＩＣとＵＳＢメモリの通信が行われていることがわかる。
⑧中身を確認するため、ＵＳＢメモリをＰＣに接続する。

以上の手順を踏めば、ＵＳＢメモリ内にＤＥＭＯ，ＦＲＯＭ＿ＭＥ，ＴＯ＿ＹＯＵという名前の３つのテキストファイルが作られる。ＵＳＢメモリをＰＣに接続し、メモ帳で確認すると、これらの内容は図７の通りになっているはずである。ＤＥＭＯ．ＴＸＴには、デモンストレーション用のテキストが書き込まれており、その他の２つのＴＸＴファイルの中身は全く同じものである。詳しい解説は後で述べるが、これはＦＲＯＭ＿ＭＥをコピー元、ＴＯ＿ＹＯＵをコピー先として、ファイルのコピーを行っているためである。

ちなみに、著者は３社のＵＳＢメモリを試したが、全てのＵＳＢメモリで正しく動作することが確認出来た。

図７、各テキストファイルの内容

６、動作の解説

ここまでは、システムの再構築をするための説明をしてきた。ここからは、どの様に３つのテキストファイルが作成されたのかを説明する。巻末の付録のｍａｉｎ．ｃには多めにコメント文をつけておいたので、比較しながら読むと、より理解が進むのではないかと思われる。

①ＵＳＢメモリのルートディレクトリにＤＥＭＯ．ＴＸＴを作成する。
②ＤＥＭＯ．ＴＸＴに次の文を書き込む

```
Hello.
Hello.
I am a MicroController.
1
2
3
```

②ＦＲＯＭ＿ＭＥ．ＴＸＴを作成し、

```
abcdefg
hijklmn
opqrstu
vwxyz
```

を書き込む。

③ＴＯ＿ＹＯＵ．ＴＸＴを作成する。

④ＦＲＯＭ＿ＭＥ．ＴＸＴの内容を読みだし、ＴＯ＿ＹＯＵ．ＴＸＴに書き込む。

ファイルの作成とその読み書きというシンプルな動作ではあるが、この３つの動作ができれば、ＰＩＣとＵＳＢメモリ間の最低限のデータ記録、及び移動処理を行うことが出来たと考えている。これらを基礎として、様々な応用プロジェクトがつくれると思われる。

７、考察と今後の展望

この研究を行っていて気がついたこと、及び、試してみたことなどを以下に述べる。

①ＵＳＢに供給している５Ｖの電圧を３．３Ｖとしたところ、問題なく動作した。エラーが生じる危険性もあるが、単一電源動作ができるという利点は大きい。

②改行したいときは、¥r¥n という２文字を用いる必要がある。¥n のみの場合、文字列を書き込んだファイルをメモ帳で開くと、↑の様な文字が表示されるだけで、改行が行われない。ある程度高機能なテキストエディタならば¥n だけでも正しく表示される。これは、マイクロソフトＯＳならではのものであると考えられるため、他のＯＳでは問題ないと思われる。

③実数の書き込みをすることができない。具体的には、ＦＳｆｐｒｉｎｔｆ関数において％ｆや％ｌｆを使用した場合、エラーなどは出ないが正しく動作しない。リファレンスマニュアルにも、実数に関する記述は見当たらなかった。

④マイクロチップ社の提供するＦＡＴファイルシステムは、ロングファイルネームに対応していない。ＬＦＮ対応のファイルシステムはフリーで公開されているので、マイクロチップ社のものと入れ替えることは出来ないか。

⑤ファイルの作成日時、更新日時などが１６０１年１月１日になってしまう。リアルタイムクロックモジュールの導入により解決できると考えられる。

⑥フォルダの作成やファイルの削除など、さらに多くの処理を実装したい。

⑦今回は４ＧＢのＵＳＢメモリを使用した。４ＧＢ以下ならば動作すると考えられるが、８ＧＢ、１６ＧＢとなると動作しない可能性がある。

⑧将来的には、ＬＣＤやスイッチなどを付けて、ＵＳＢメモリ内のファイルを操作する１つの機器にまとめたい。製作例として役割は、この論文が果たしていると考えてはいるが、ＵＳＢメモリを制御しているという実感があまり湧かないというのが本音としてある。もう少し高度な処理を加えてモジュール化できれば、展示用としても良いものが出来るのではないか。

８、ソフト入手先・参考文献

［１］Microchip Technology Inc. - http://microchip.com

［２］PIC24FJ64GB004 Family Data Sheet（Microchip Technology Inc.）

［３］Implementing File I/O Functions Using Microchip's Memory Disk Drive File System Library（Microchip Technology Inc. Application Note 1045）

２０１０年　１０月１３日