USB メモリ・TFT 液晶・タッチパネルを用いたシステムの構築

Construction The System Using Thumb Drive, TFT-LCD, and Touch Panel

独立行政法人国立高等専門学校機構

Institute of National Colleges of Technology

小山工業高等専門学校　電子制御工学科

Oyama National College of Technology

Department of Electronic Control Engineering

金野研究室

Kinno Laboratory

本澤　上

Joe Honzawa

まえがき

本書は，USB メモリ，TFT 液晶，タッチパネルを使用した機器の製作過程をまとめたものである．ある程度の電子工作の経験と，PIC マイコンを使った経験がある人ならば，本書を読みながらシステムを構築できるように留意して書いている．また，電子工作にあまり馴染みの無い人も読みやすいように，可能な限り詳しく解説を入れている．そのため人によっては説明がくどいと思われるかもしれないがご容赦願いたい．

また，卒業論文という形をとっているが，むしろ製作記録と言ったほうが適切であると思っている．そのため，いわゆる ”正しい論文の書き方 ” には準拠させていない部分が数多くある．この点も合わせて，寛容な心で読んでいただきたい．

2011 年 2 月 11 日　筆者

目次

1　はじめに

世間で見かける携帯型機器，例えば携帯電話や音楽プレイヤー等を見ると，ほぼ 100 %にタッチパネルが採用されている．タッチパネルが利用できれば，機器に取り付けるスイッチ類の数が格段に減ることは明らかである．現に，操作のほぼ全てをタッチパネルから行う携帯端末も発売されている．

このタッチパネルには多くの方式があり，特徴も様々である．その中で手に入り易く，かつ安価なものに 4 線抵抗膜方式がある．電子工作レベルでのタッチパネルの活用を目指すならば，これを用いるべきであろう．

本研究では，タッチパネルの制御をするとともに，大容量化が進む USB メモリからのデータ読み出しを行う機器の製作を目的とする．

2　概要

図 1 に，システムの概要図を示す．全体の制御には，Microchip 社の PIC マイコンを使用する．型番は PIC24FJ128GB106(64 ピン TQFP)である．TFT 液晶とタッチパネルは，電子部品のインターネット通販事業を行う aitendo@shopping より購入できる，2.4 インチ TFT 液晶とタッチパネル，そしてキャリーボードが一体となった，YHY024006A-PCB を利用する．この液晶にはフレームバッファが搭載されているため，マイコンとの相性がよく，キャリーボードが付属しているため配線の引き出しが容易となっている．表 1，表 2 に，PIC マイコンと TFT 液晶の仕様・性能を示す．

具体的な動作は，USB メモリ内をブラウジングし，選択した画像ファイルを画面に表示する. 表示する画像は bmp 形式とし, 画像の大きさは液晶のサイズに合わせて, 高さ 240 ピクセル，幅 320 ピクセルとする．なお，ビットの深さは RGB 各 8bits の 24bits とする．

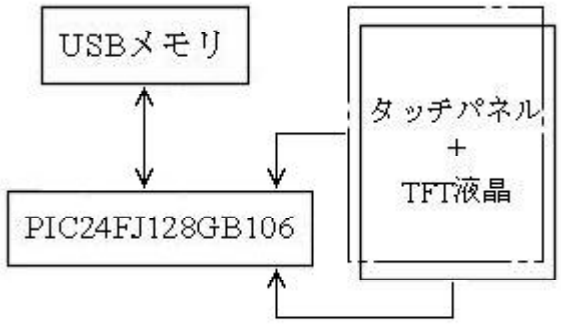

図 1　システム概要図

表 1　PIC24FJ128GB106 の性能

		PIC24FJ128GB106
ピン数		64
プログラムメモリ		128 KB
SRAM		16 KB
最高動作周波数		32 MHz
ｱ(ｻGf	ピン数	29
	16ビットタイマー	5
	入力キャプチャ	9
	PWM出力	9
	UART	4
	SPI	3
I^2C		3
10ビットA/D変換		16 ch
コンパレータ		3
PMP/PSP		○
JTAG		○
CTMU		○
USB On-The-Go		○

表 2　YHY024006A の仕様

		YHY024006A
画面サイズ		2.4 inch
解像度		240×320
インターフェース		16 bits(R:5bits, G:6bits, B:5bits)
電源電圧		3.2 V
c°"\;	順方向電圧	3.2 V
	順方向電流	60 mA

3　開発環境

参考までに，著者の開発環境を示す．基本的に，バージョンの違いによる差異はそれほど大きなものではない．コンパイルが通る通らないという差がでることはまず無いであろうし，バージョンの違いによって不可能となるようなことは特にしていない．しかし，どうしても再構築が上手くいかない，出来ないという場合はバージョンを合わせてから挑戦して頂きたい．

・統合開発環境：MPLAB IDE v8.46
・コンパイラ：MPLAB C30 C Compiler v3.23
・PIC ライタ：Microchip MPLAB ICD 2
　ICD 2 version：8.43.02.02
　Firmware version：3.04.08.00
　Bootloader version：1.01.01.00
・OS：Microsoft Windows XP Professional Service Pack 3

4　原理

4.1　TFT 液晶

TFT 液晶の原理について簡単に説明する．まず TFT とは，Thin　Film　Transistor の略であり，TFT 液晶は薄膜トランジスタ液晶と訳される．実際に液晶を見ると，1 画素に RGB の 3 本の線が見受けられるが，この 1 本をトランジスタ 1 つが制御している．すなわち，画素 1 つに対して 3 つのトランジスタが存在している．各色の明るさは，各トランジスタのベース電流に比例する．なお，3 原色がライン状に配置されておらず，RGB のドットが並んでいるものなど様々種類がある．

4.2　タッチパネル

本装置に使用した 4 線抵抗膜方式タッチパネルの原理を説明する．基本的には，図 2 に示すように 2 枚の抵抗膜が重ねられた構造となっている．それらの間にはドットスペーサと呼ばれる点状のスペーサが規則的に配置されており，それぞれの抵抗膜には電極が 2 本ずつ取り付けられている．

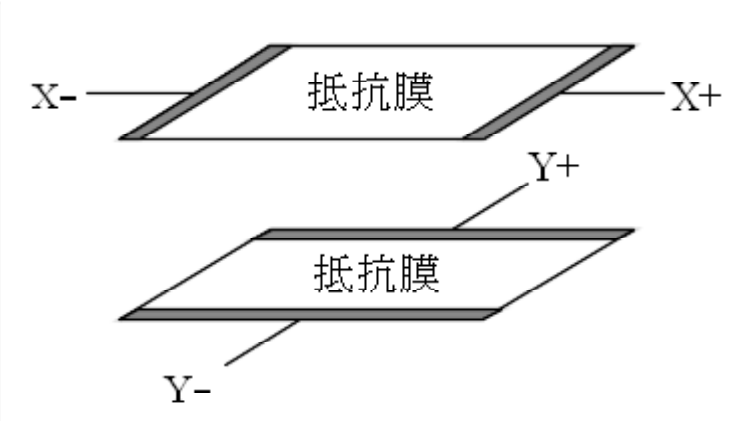

図 2　4 線抵抗膜方式タッチパネルの原理

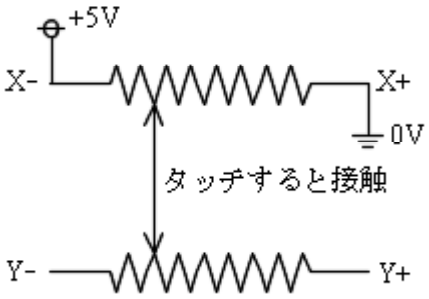

図 3　座標検出原理

Y 座標検出の方法について以下に述べる．まず，X-に電源，X+にグランドを接続する．ユーザーがパネルに触れると抵抗膜がたわみ，2 枚が接触する．図 3 のように抵抗膜が接触すると，Y+，Y-に分圧された電圧が現れ，その値はタッチ点の Y 座標によって決定される．これにより，ユーザーがタッチした点の Y 座標を検出することが出来る．

マイコンでこれを行う場合は，I/O ポートから X-，X+に H レベルと L レベルを与えて，Y+または Y-を A/D 変換することになるが，ここで注意するべき点がある．A/D 変換を行う際には，Y+と Y-に接続するピンは全てハイインピーダンス状態でなければならない．一般的にはアナログ入力モードとしておけば問題はないが，仮にデジタル入力モードなどに設定されていた場合，そのピンに電流が流れ込むために正しい電圧を読み取ることが出来ない．

5　ハードウェア

5.1　回路図

図 4，図 5 に，システムの回路図を示す.

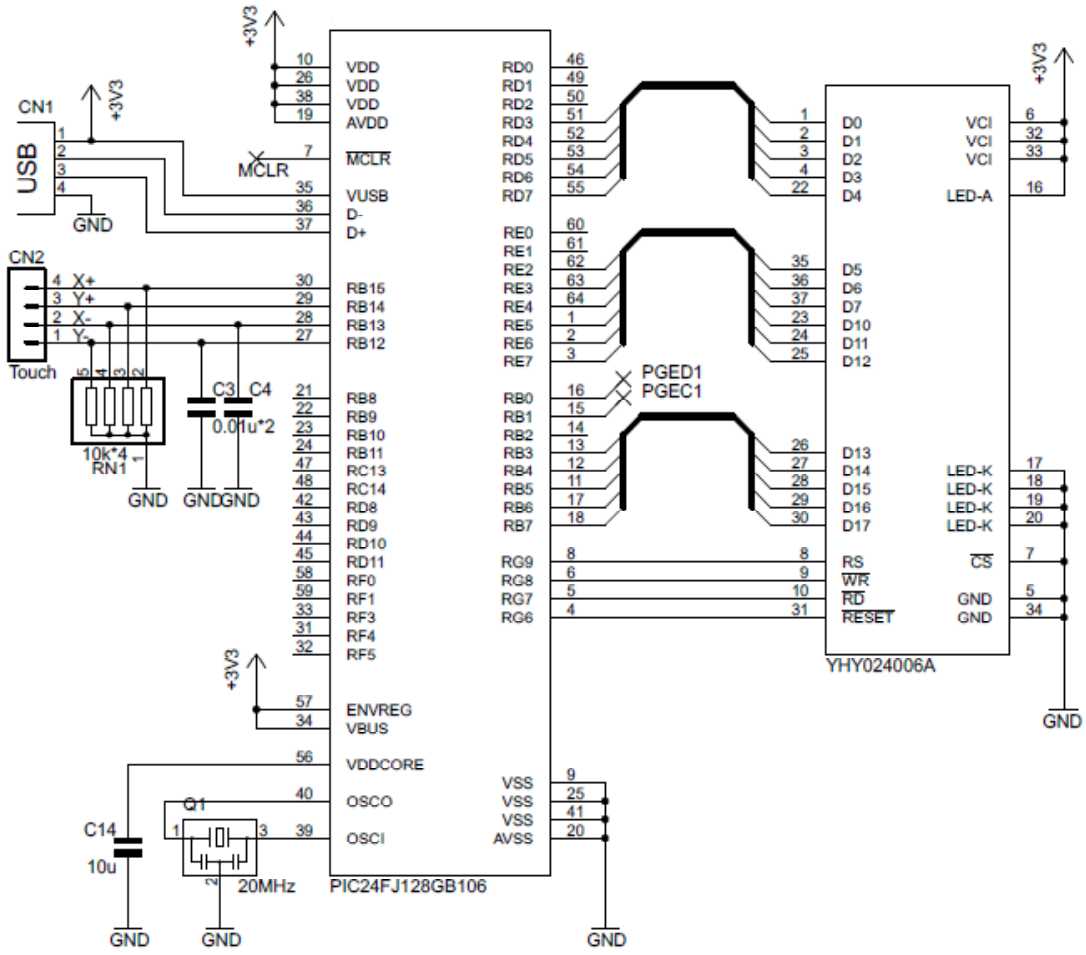

図 4　システム回路図(メイン部分)

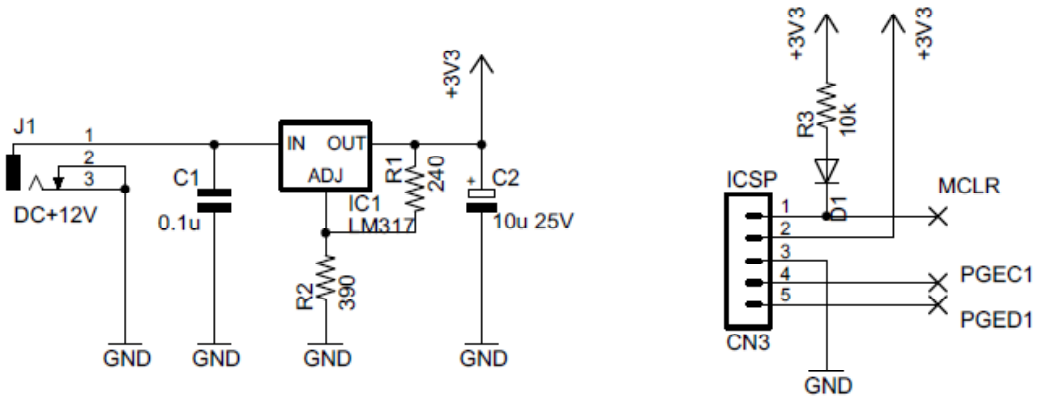

図 5　システム回路図(電源回路と ICSP 回路)

まず図 4 のメインの回路について説明する．中心にある IC が，全体の制御を行う PIC マイコン"PIC24FJ128GB106"である．この PIC は TQFP パッケージであるため，一度ピッチ変換基板で 2.54mm ピッチに直している．左上の VUSB，D-，D+ピンが USB メモリと直接やりとりをするピンである．また，USB 機能を有効にするためには内部レジスタ等の設定も必要だが，ハード面でも 34 番ピンの VBUS を電源へ接続する必要がある．本来ならば，データシートの例のように回路の保護なども考えるべきであるし，USB の規格に従って VBUS には 5V を供給するべきであるが，問題なく動作しているため今回は割愛した．なお，USB 機能を使用したい場合は，PIC に入力するクロックは 4MHz の倍数でなければならない．PIC 内部では，入力したクロック(今回は 20MHz)を一旦 4MHz まで落として，PLL によって 96MHz まで上げてから，それを基準に 48MHz をつくって USB メモリとの通信に利用している．CPU の動作クロックとして使用される周波数 T_{OSC} は最高で 32MHz(96MHz ÷ 3)であるため，1 命令にかかる時間 T_{CY} は，次式で表される．

$$T_{CY} = \frac{2}{T_{OSC}} = 62.5\,\mathrm{ns} \tag{1}$$

RB12 から RB15 は，タッチパネルとのやり取りに使用する．非タッチ状態の応答を安定させるために 4 線全てを 10k Ωでプルダウンしている．また，X-と Y-に接続されているコンデンサは，ノイズの影響を軽減する目的で付けている．4 本の配線にはアナログの信号が通るため，可能な限りデジタル線からは離したほうがよい．しかし，前述したように使用する PIC が TQFP パッケージであるために，結局は距離をおくことができない．この問題はソフトウェア側で対処するべきである．この点については後に述べる．

次に図 4 の右側の，液晶との接続について述べる．USB メモリから読み込むビットマップ画像は，RGB の各色が 8 ビットで表されるが，YHY024006A のインターフェースは RGB がそれぞれ 5bits，6bits，5bits に設定されているので，読み込んだデータの下位ビットをいくつか切り捨てて送信する必要がある．その度にビットシフトを繰り返すのは効率が悪いので，ポート D，E，B をそれぞれ RGB に対応させ，上位ビット側に詰めて配線してプログラムの簡単化及び高速化を図っている．これらデータ線の他に，制御線が 5 本存在する．それらの意味を次の表にまとめる．

表 3　制御線の名称と意味

CS	チップセレクト端子 - L レベルでデバイスを有効にする.
RS	レジスタセレクト端子 - コマンド送信の際はL レベル，データ送信の際はH レベルとする.
WR	ライト端子 - 書き込みの際にストローブ(H→L→H)する.
RD	リード端子 - 読み込みの際にストローブ(H→L→H)する.
RESET	リセット端子 - L レベルにするとデバイスがリセットされる.

表中の端子名には無いオーバーラインが回路図の端子名に付いているが，これは端子が負論理(L レベルで ON，H レベルで OFF)であることを示すものである．まず始めに、CS 端子は同じ通信方式のデバイスを複数個制御する場合に使用する．1 個 1 個にそれぞれデータ線を配線するのは，マイコンのピンを無駄に使い効率が悪い．データ線は全て共通とし，現在送信している信号がどのデバイスへ向けたものなのかを示す信号をそれぞれに配線することが望ましい．つまり CS 端子が H レベルの間は待機し、L レベルのときのみ信号を受信するのである．今回は液晶を 1 つしか使用しないので，CS 端子は常に L レベル，すなわち GND 電位で問題ない．RS 端子は，送信する信号にコマンドまたはデータとしての意味を付加する端子である．送信する信号に合わせて適宜制御する必要がある．WR 端子、RD 端子はそれぞれ書き込み，読み込みの際に使用する．今回は読み込む処理はしていないので，RD 端子は H レベルに固定となっている．RESET 端子は，その名の通りデバイスをリセットする端子である．今回は電源投入後の液晶の初期化の際にしか使用していない．これらの端子名は，他の種類のデバイスにもよく見られるものであるが，その意味が必ずしも表 3 と同一とは限らない．各々のデータシート，テクニカルノートを確認することを強く勧める．

図 5 の回路について述べる．左側は，3 端子レギュレータ LM317 を用いた 3.3V の電源である．特別な工夫は特に無い．出力される電圧は 2 つの抵抗 R_1，R_2 によって決定される．その式を以下に示す．

$$V_{out} = 1.25\,V \times \left(1 + \frac{R_2}{R_1}\right) \tag{2}$$

この式で計算すると，出力は約 3.28V となる．回路図では入力電圧が 12V となっているが，6V 程度でも問題はない．

右側の 5 本の端子は，基板上で PIC にプログラムを書き込むための端子である．一般に ICSP(In-Circuit Serial Programming)と呼ばれている．各端子は，1 番から順に MCLR，VDD，GND，PGEC，PGED と並んでいる．PGEC，PGED ピンが PIC 以外に接続されている場合は，書き込みの際に分離できるようにスイッチなどを設けるべきである．書き込みを行う際は回路側から電源を供給する必要がある．ライタは，Microchip 社純正品の使用を勧める．

5.2　パターン図

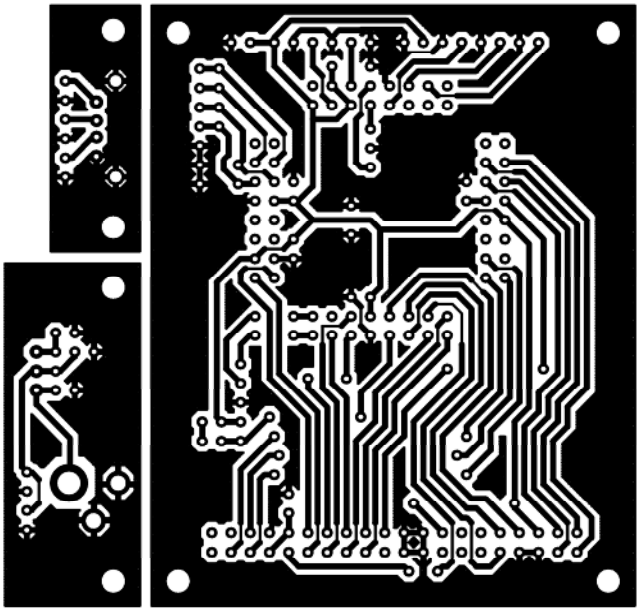

図 6　パターン図

図 6 にパターン図を示す．ケースへ実装することを考えて，本体基板，USB 基板，電源基板の 3 枚に分割している．次項に示す写真のように，ピンヘッダなどを使用して配線している．

5.3　回路写真

以下に回路の写真をいくつか示す．注意してほしいのは，写真にある回路のパターン図は一度製作したものを手直ししたものであるため，図 6 のパターン図とは若干異なる部分があるという点である．その点については以下で写真と共に説明する．

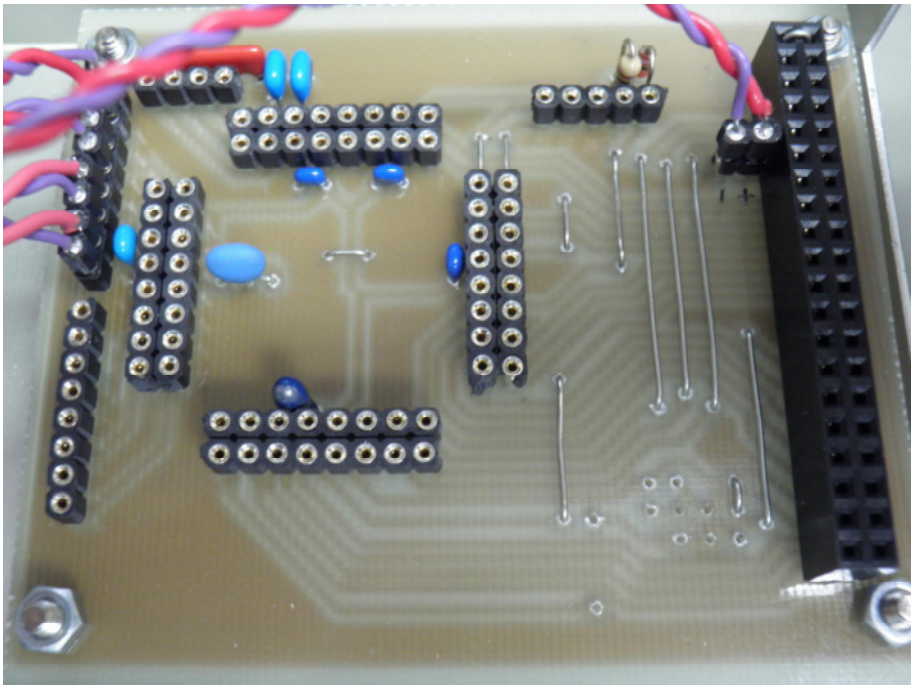

図 7　メイン基板

図 7 の右下辺りに，パターン図には無い穴とジャンパ線が存在している．

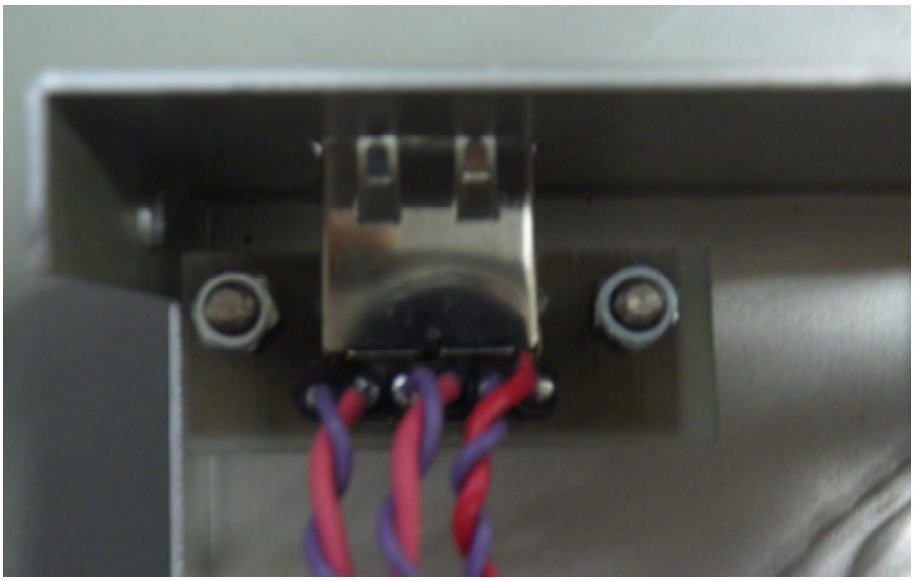

図 8　USB 端子基板

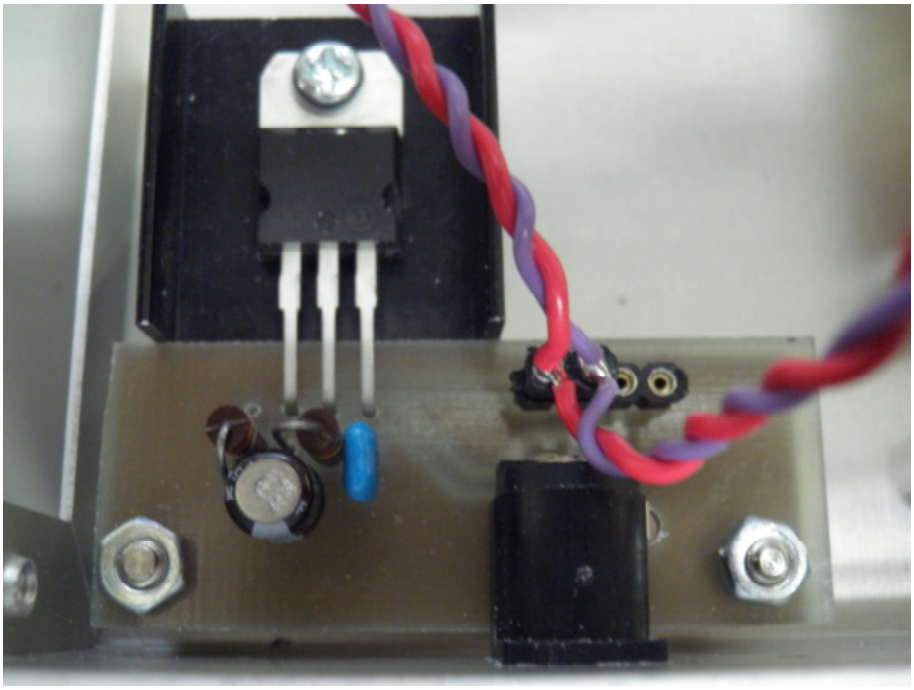

図 9　電源基板

当初は 12V も使用する予定であったため，そのためのピンソケットを用意していた．図 6 のパターン図では，どちらも 3.3V に接続されている．

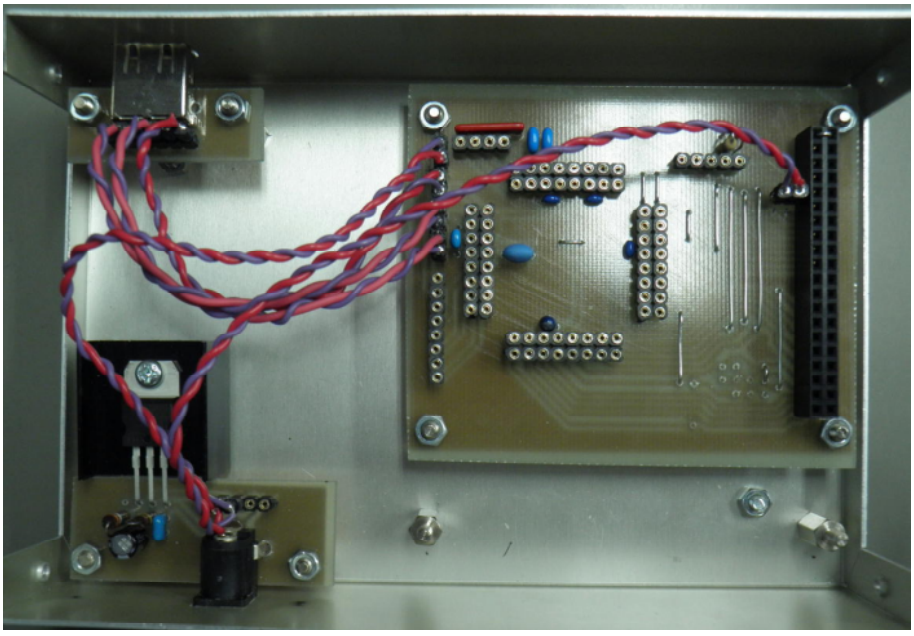

図 10　回路の接続

図 10 に示すように，回路と回路の接続は，ツイストペアにした導線をピンヘッダに半田付けした簡易のケーブルで行っている．そのため，ピンヘッダは 1 穴置きにグランドへ接続されている．しかし，デジタル信号が乱されるような強いノイズが乗ることは稀であるため，無理にツイストペアにする必要はない．

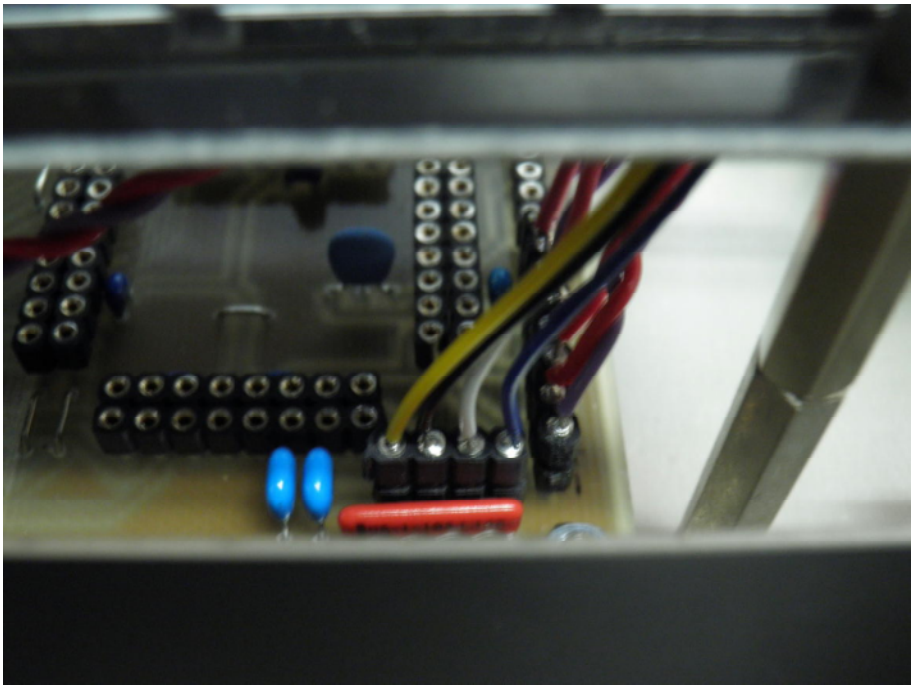

図 11　タッチパネルの接続

図 11 に，タッチパネルとの接続の様子を示す．ケーブルは YHY024006A 付属のものを使用した. 図の方向から見たときに, 黄, 黒, 白, 青の順に接続する. 端子名で言えば Y-, X-, Y+，X+の順である.

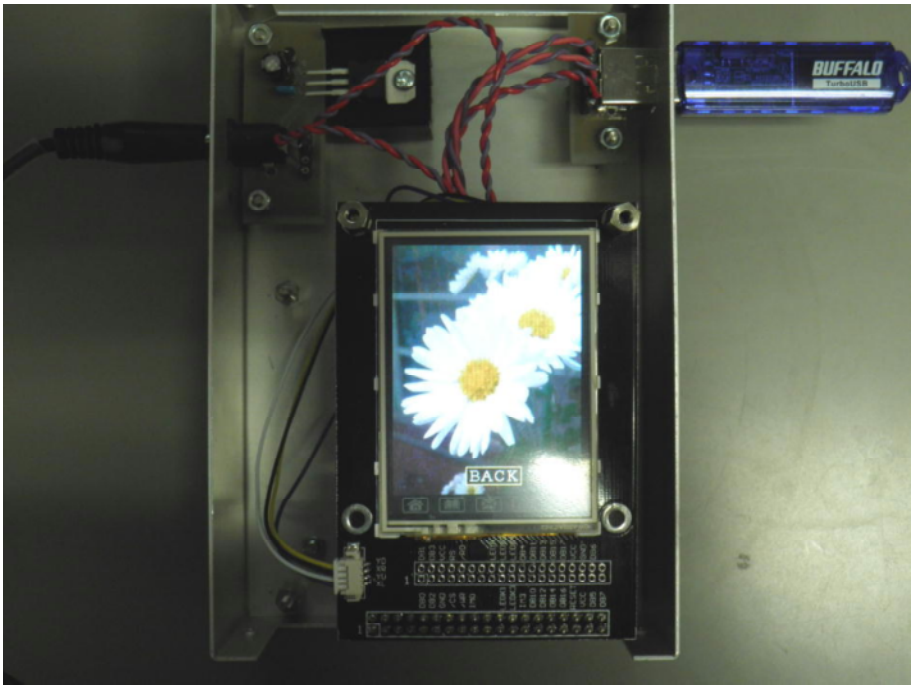

図 12　動作の様子

6 ソフトウェア

6.1 USB メモリ

USB メモリとの通信，及び FAT ファイルシステム関連の処理は，Microchip 社から提供されているサンプルを使用した．具体的には，サンプル集をインストールすると作成されるフォルダ"Microchip Solutions"内の"USB Host - Mass Storage - Simple Demo"を参考としている．同様のサンプルに"USB Host - Mass Storage - Thumb Drive Data Logger"があるが，こちらはやや複雑であったためにあまり参考にはしていない．

Microchip 社の FAT ファイルシステムは，FAT12，FAT16，FAT32 の全てに対応しており，PIC18 や dsPIC などでも幅広く使用できる．しかしロングファイルネーム(LFN)には対応しておらず，8.3 形式のファイル名しか扱えないという欠点がある．

6.2 TFT 液晶(YHY024006A)

液晶へのデータ、及びコマンドの送信(正確にはレジスタへの書き込み)について解説する．TFT 液晶への信号の規約は YHY024006A のデータシートに記載されてはいるが，内容が乏しくデータシートを読んだだけではまともに制御することは出来ない．そのため，液晶のコントローラチップである ILI9325 のデータシートを参考としている．通信方式やレジスタの情報などが細かく記載されており，非常に有用な情報源である．今回使用した通信方式は，ILI9325 のデータシート内では i80 と称されているものである．注意しなくてはならないことは，IM ピンによって通信方式が選択できると書いてあるが YHY024006A では i80 に固定されているということである．また，この液晶を取り扱っているネット通販サイト"aitendo"では，この液晶制御の C 言語サンプルを公開している．初期化手順など，非常に参考になるサンプルである．

次に，文字の表示について述べる．表示可能とした文字は，必要最低限の英数字と記号のみ(ASCII コードの 0x20 から 0x7E まで)である．本来は半角フォントを用いるべきであるが，フォントの著作権などの都合から 16bits × 16bits の全角文字を使用している．フォントデータはヘッダファイル ascii.h に記述してある．

6.3 タッチパネル

タッチパネルの処理について説明する．上述した原理が基本であるが，ソフトウェア的にノイズの影響を抑えるため，プラス端子とマイナス端子の両方からサンプリングして平均をとり，更にそれを 32 回行って，それらの平均を最終的なサンプル値としている．つまり，タッチ点の判定 1 回につき計 64 回のサンプリングを行っている．しかしこれだけでは，タッチの強弱によっては誤った点を検出してしまうため，タッチ点判定を 2 回行い，X 座標，Y 座標ともに同一の値となったときに値を採用している．

次に座標補正について述べる．単に A/D 変換結果の平均をとっただけでは，タッチされた点を求めることは出来ない．少なくとも 2 点は，画面上の座標と A/D 変換値の対応がわかっているポイントが必要である．今回は電源投入後に，(80,100)，及び(200,250)に 1 ドットの白い点を表示して，その点をタッチしてもらうという手段をとった．得られた電圧値(X_{80},Y_{100})と(X_{200},Y_{250})から，まず画面端をタッチした際に得られる電圧値を推定する．

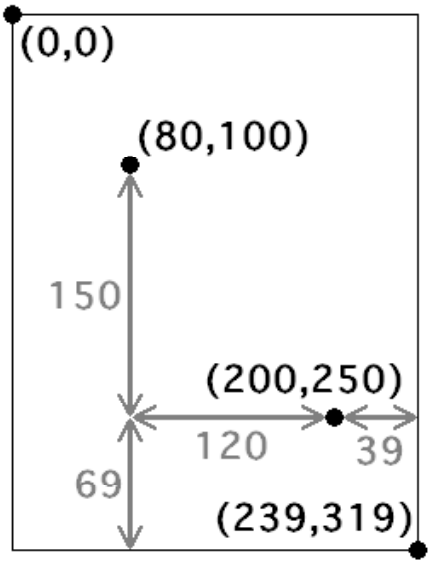

図 13　座標補正

図 13 を見れば，X 方向，Y 方向の 1 ピクセル当たりの電圧の変化量は，

$$\begin{cases} \Delta V_{\mathrm{X}} = \dfrac{X_{200} - X_{80}}{120\ \mathrm{pxl}} \\ \Delta V_{\mathrm{Y}} = \dfrac{Y_{250} - Y_{100}}{150\ \mathrm{pxl}} \end{cases} \tag{3}$$

であることがわかる．なお，画素の単位(ピクセル)の表記は pxl とした．式(3)を用いれば，画面の左端，右端をタッチした際の X 座標を示す電圧 X_0，X_{239}，及び，画面の上端，下端をタッチした際の Y 座標を示す電圧 Y_0，Y_{319} は，次の式(4)で表される．

$$\begin{cases} X_0 = X_{80} - (80\ \mathrm{pxl}) \times \Delta V_{\mathrm{X}} \\ X_{239} = X_{200} + (39\ \mathrm{pxl}) \times \Delta V_{\mathrm{X}} \\ Y_0 = Y_{100} - (100\ \mathrm{pxl}) \times \Delta V_{\mathrm{Y}} \\ Y_{319} = Y_{250} + (69\ \mathrm{pxl}) \times \Delta V_{\mathrm{Y}} \end{cases} \tag{4}$$

さらにこれを用いて，A/D 変換値(x_{AD},y_{AD})からピクセル座標(x,y)への変換式(5)が得られる．

$$\begin{cases} x = (239\ \mathrm{pxl}) \times \dfrac{y_{\mathrm{AD}} - X_0}{X_{239} - X_0} \\ y = (319\ \mathrm{pxl}) \times \dfrac{y_{\mathrm{AD}} - Y_0}{Y_{319} - Y_0} \end{cases} \tag{5}$$

7　使用方法

以下に，本装置の使用法，及び操作方法を示す．わかりやすいように，具体的なファイル，フォルダを例に説明する．

7.1　ファイルの準備

まず，USB メモリに表示する画像ファイルを準備する．画像ファイルが満たすべき要件は,

・拡張子が bmp または BMP であり，ビットマップフォーマットに従っていること．

・1 ピクセルが RGB 各 8bits で表されていること．

・画像サイズが幅 240 ピクセル，高さ 320 ピクセルであること．

・ascii.h(付録参照)に含まれていない文字がファイル名に使用されていないこと．

である．上の 2 つは, 画像をペイントソフトで編集, 保存する際に指定することができる．確認したい場合は，画像のプロパティ→概要タブ→詳細設定と進み，幅，高さがそれぞれ 240 ピクセル, 320 ピクセル, ビットの深さが 24 となっているか確かめればよい. また, USB メモリ内のフォルダが満たすべき要件は,

・ascii.h(付録参照)に含まれていない文字がファイル名に使用されていないこと．

である．当然，改良を加えればこれらの要件を必要としないファイル，フォルダに対応させることは可能である．以下で例として使用するファイル，フォルダの構造を図に示す．

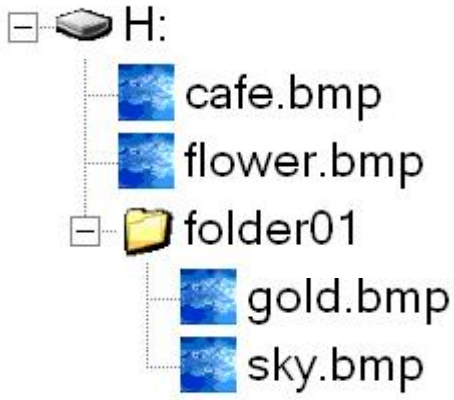

図 14　ファイルの配置

7.2　電源投入後の流れ

画像ファイルが保存された USB メモリを本体の USB 端子に接続してから，電源プラグを接続する．はじめに画面が白く光り，その後黒く塗りつぶされ，キャリブレーションが始まる．順次表示される 2 点を正確にタッチすると，その後 USB メモリのルートディレクトリ(1 番上の階層)内のファイルとフォルダが表示される．この例では CAFE.BMP, FLOWER.BMP，FOLDER01 が表示される．それぞれ名前の部分をタッチして選択し，画面下部に表示されるボタンに触れることで, 画像の表示やフォルダ間の移動が可能である．なお，ルートディレクトリ以外では，存在するフォルダの他に「..」というものが表示されるが，これは「1 つ上のフォルダ階層」という意味であるので，これを開くと上の階層へ戻ることができる．FOLDER01 内にはフォルダは存在しないが，「..」が表示されるは

ずである．以下に，動作中の写真を数点示す．

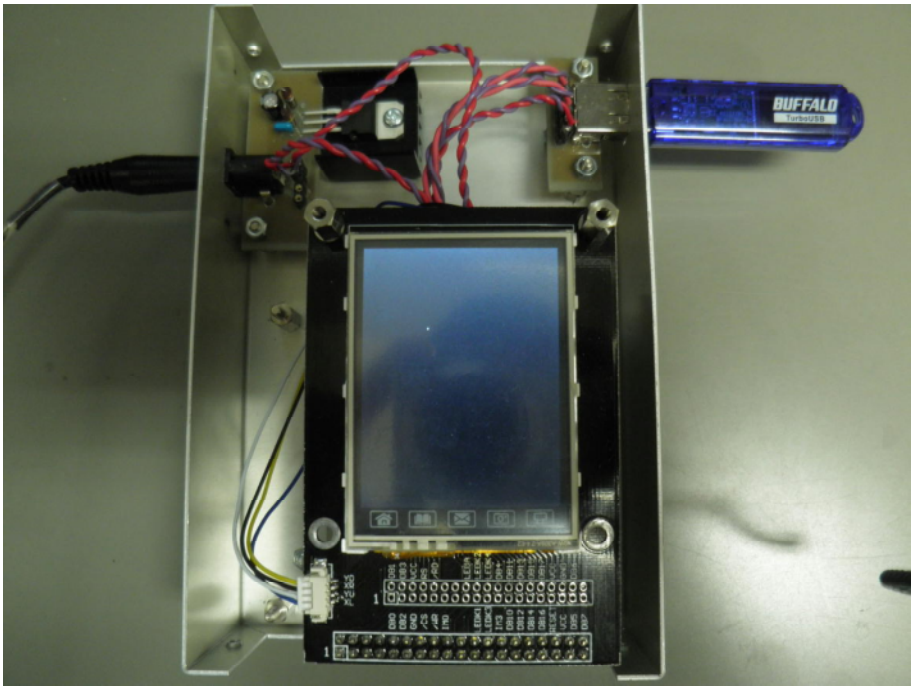

図 15　キャリブレーション動作

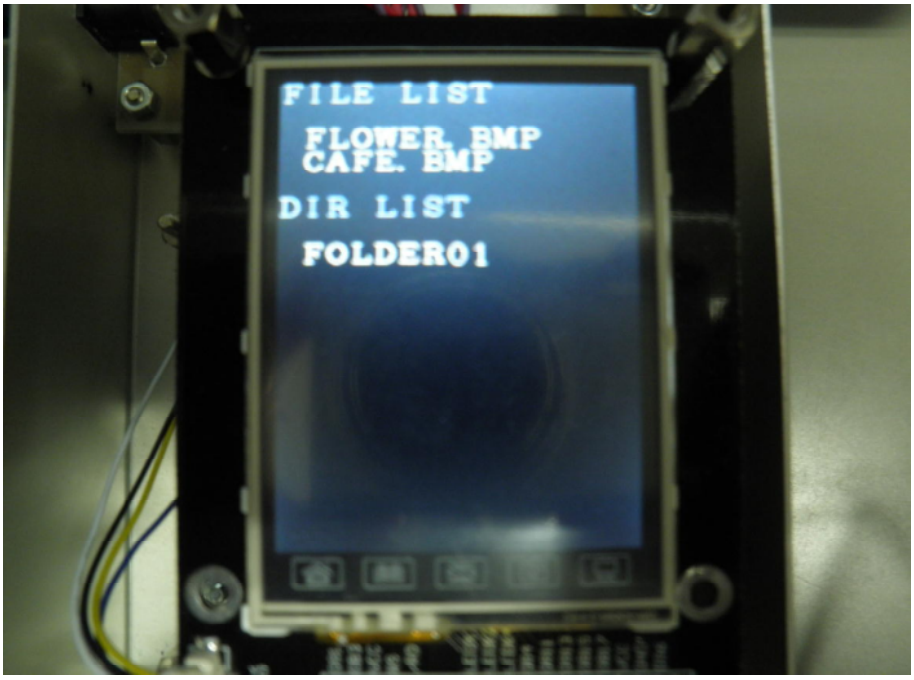

図 16　ルートディレクトリの表示

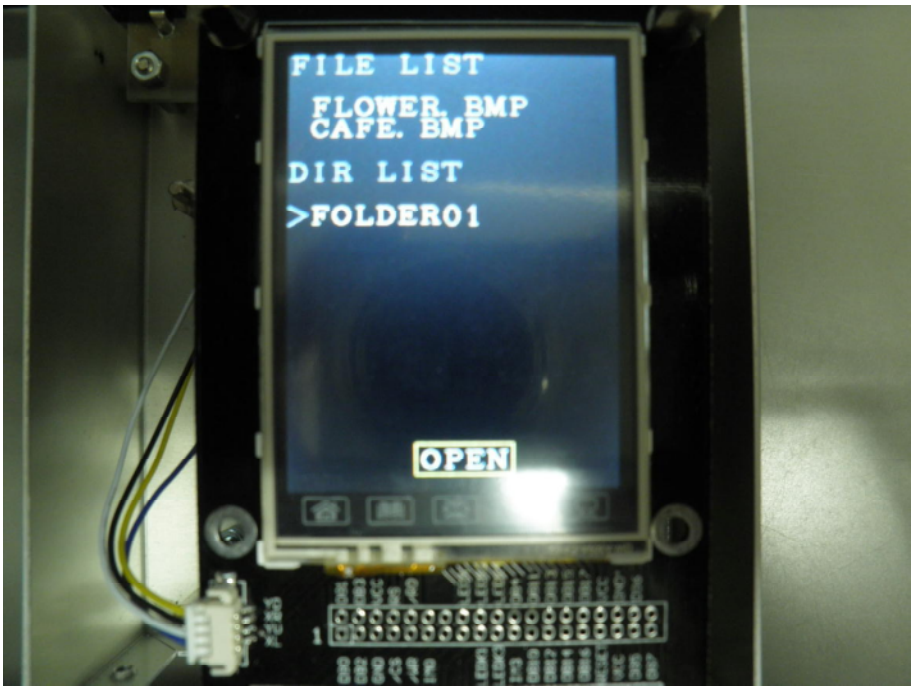

図 17　FOLDER01 を選択した様子

図 18　FOLDER01 の内部

図 19　GOLD.BMP を選択した様子

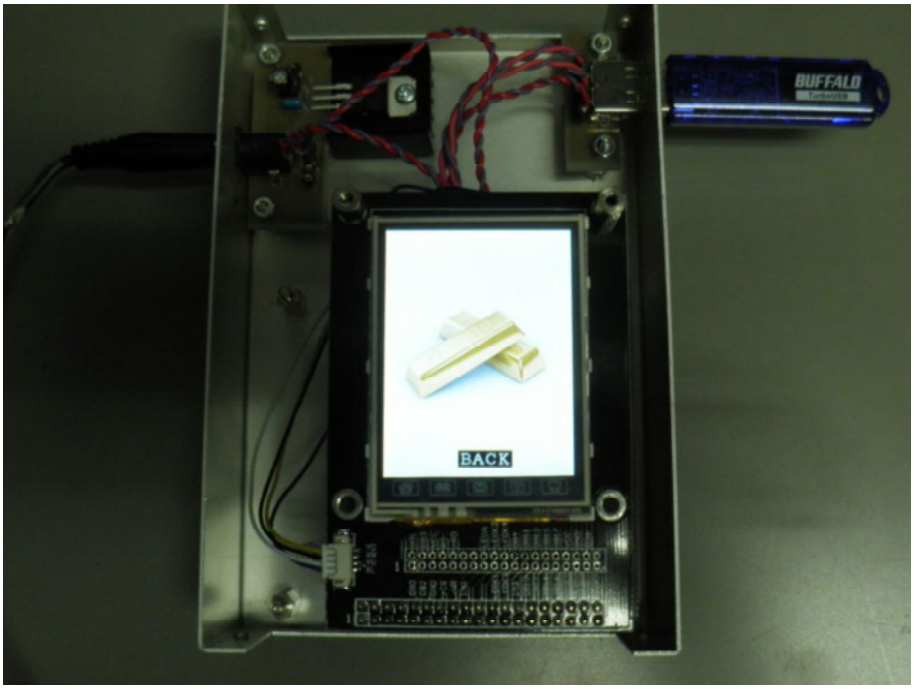

図 20　GOLD.BMP を表示した様子

7.3　電源の切断

電源を切断するときに特別な手順は無く, 電源のプラグを引き抜いてしまって問題ない. ただし，画像を表示し始めてから描き終わるまでの間，すなわち画像を描いている最中は電源を切ってはならない．USB メモリ自体や，そのファイル構造を破壊する恐れがある．

8 考察

8.1 現状と問題点

現時点で挙げられる問題点を順に示す.

ⅰ） まず USB 関連の処理であるが，基本的な処理は全て Microchip 社のライブラリに用意されているため，ファイル操作には問題点は見当たらない．しかし，LFN(ロングファイルネーム)，及び 2 バイト文字への対応があればなお良い．この点は，以前 USB メモリへの読み書きを目的として製作を行ったときにも問題点として挙げていたが，有効な対策を講じることが出来ずにいた.

ⅱ） 次に液晶について述べる．今回の液晶は 240 × 320 の解像度を持つものであった．つまり，1 画面の更新に 76800 ピクセルの情報を送信しなければならない．画面を黒く塗りつぶす際はデータバスの出力を 0x00 に固定して高速化を狙ったが，それでも 0.2 秒ほど時間がかかり，肉眼で画面の書き換えがわかってしまう．また，画像の描画の際は USB との通信作業もあるため，およそ 2 秒の時間を要する状況である.

ⅲ） 現在は，表示可能な文字は英数字とある程度の記号のみであるが，やはり実用する上で望まれるのは日本語への対応である.

ⅳ） タッチパネルの処理には，致命的な欠陥がある．抵抗膜の抵抗値分布が一様なタッチパネルであれば一切問題はないが，少しでも分布に歪みのあるタッチパネルを扱う場合は式(3)，(4)，(5)で表される計算では不十分である．式を見れば明らかであるが，このキャリブレーション法は，タッチ座標と出力電圧の関係が線型であることが前提となっているのである．実用上問題のない程度ではあるが，筆者が使用したタッチパネルにも若干の歪みが見られた．抵抗値分布が完全に数式で表現されるか，または何らかの形で明示されない限り，100%の補正を行うことは不可能であるが，画面をいくつかの領域に分割して検出・補正を行うことである程度解決できると考えられる.

8.2 解決策

ⅰ） 前項で挙げた問題点に対する解決策を述べる．LFN，2 バイト文字への対応を実装するためには，Microchip 社の FAT ファイルシステムライブラリに直接手を加えて拡張する必要がある．そのためにはライブラリ全体をよく把握し，既存の処理とバッディングしないように新機能を実装しなければならず，筆者の力量では到底出来そうもない．そこで考えられるのが，別のファイルシステムの利用である．フリーのものでは，ChaN 氏によって製作された"汎用 FAT ファイルシステムモジュール Fat-Fs"がある．こちらのライブラリは LFN と Unicode に対応しているため，前述の問題点を一挙に解決出来る．加えて，リード・オンリー構成のシュリンク版も用意されており使い勝手がよい．しかし，以前こちらを移植して(既存の変数型との衝突を解決する作業がほとんどであった)動作させることには成功していたが，ここでも問題が見られた．正しく動作する USB メモリと，そうでないものが現れてしまったのである．また，ほとんどの USB メモリにある，通信中を

示す LED の点滅が見られなかった．原因として挙げられるのは，互換性である．Microchip 社のライブラリは，USB 通信もファイルシステムも全て Microchip 社が製作したものであるために相互の互換性があったが，Microchip 社の USB 通信ライブラリと ChaN 氏のファイルシステムライブラリを併用したため，何らかの不具合が生じたと考えられる．移植の方法によってはこの不具合を解消できると思われる．仮にそれが成功すれば，USB 関連の処理はほぼ完璧なものとなると考えられる．

ii） 液晶の描画速度の向上を目指す場合，より高速動作できるマイコンの使用が考えられる．例としては，PIC32MX シリーズがある．このシリーズは，USB 通信用のクロックは PIC24F シリーズと同じく 48MHz までであるが，CPU 動作用のクロックは 80MHz まで上げられるため，現在の約 2.5 倍での動作が見込める．静止画の表示に限った動作には十分な速度であるが，それだけ高速に動作すれば，消費電力など別の問題が浮上することが考えられる．別の案としては，USB メモリよりも高速にデータの読み出しが可能なメディアを外付けメモリとして設置することが挙げられる．画像データの読み出しにかかる時間が短縮されれば，画像描画も当然速くなる．なお，動画などのさらに高速な描画を目標とする場合は，もはや PIC マイコンからの液晶制御は無理であるため，FPGA などの IC を用いる必要があると考えられる．

iii） 常用とされる漢字は 2136 字あり，それに加えて，ひらがな，カタカナ，英数字，記号がある．これら全てを 16bits × 16bits の文字で表示可能とするならば，およそ 75KB のメモリ容量を必要とする．この全てを PIC24FJ128GB106 のプログラムメモリに書き込むことは不可能であるため，外部 EEPROM を設けて逐一読みだして文字を表示する必要があると思われる．

iv） タッチパネルの座標補正方法について解説する前に，このアルゴリズムに必要な，三角形による点の包含判定について述べる．基本的な考え方を図 21 に示す．

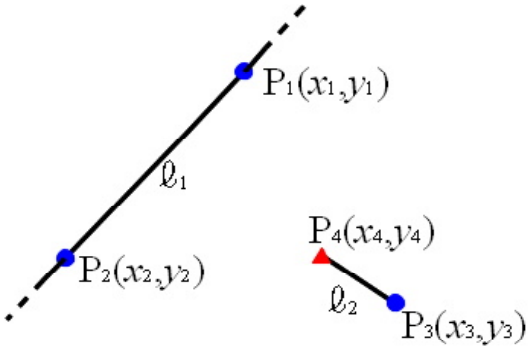

図 21　三角形による点の包含判定

まず，三角形を構成する 3 つの頂点 $P_1(x_1,y_1)$，$P_2(x_2,y_2)$，$P_3(x_3,y_3)$の内の 2 点(左図では P_1 と P_2)を結ぶ直線ℓ_1と，残りの 1 点(左図では P_3)と包含を判定する点 $P_4(x_4,y_4)$とを結ぶ線分ℓ_2 を考える．この直線と線分が交差している場合，点 P_4 は三角形の外側に存在することになる．組み合わせは全部で 3 通り存在するので，直線と線分の交差を 3 回判定すれば，点 P_4 が三角形の内部に包含されているか判断できる．直線と線分の交差判定法を，左図を用いて説明する．　直線ℓ_1を式で表すと，

$$y-\frac{y_1-y_2}{x_1-x_2}x-y_1+\frac{y_1-y_2}{x_1-x_2}x_1=0 \tag{6}$$

という形になる．直線上の点の座標を与えれば，必ず式(6)が満足される．一方，直線上に無い点の座標(x',y')を与えた場合は，式(6)の等号を不等号に置き換えて表現することが出来る．すなわち，(x',y')が直線より上の領域に含まれる場合は

$$y'-\frac{y_1-y_2}{x_1-x_2}x'-y_1+\frac{y_1-y_2}{x_1-x_2}x_1>0 \tag{7}$$

であり，直線より下側の領域に含まれる場合は

$$y'-\frac{y_1-y_2}{x_1-x_2}x'-y_1+\frac{y_1-y_2}{x_1-x_2}x_1<0 \tag{8}$$

である．これにより，直線ℓ_1の式に線分ℓ_2の 2 端の座標を代入したとき，その 2 つの値が異符号であれば交差しており，同符号であれば交差していないと言える．直線の方程式を変形して，

$$(x_1-x_2)(y-y_1)-(y_1-y_2)(x-x_1)=0 \tag{9}$$

と書けば，線分の端点座標を代入したときの値をそれぞれ

$$\begin{cases}(x_1-x_2)(y_3-y_1)-(y_1-y_2)(x_3-x_1)=\alpha\\(x_1-x_2)(y_4-y_1)-(y_1-y_2)(x_4-x_1)=\beta\end{cases} \tag{10}$$

と置いて，

$$\alpha\beta<0 \tag{11}$$

となった場合，直線と線分は交差している．この交差判定を 3 回行うことで，三角形による点の包含判定を行うことが出来る．なお，α β が 0 に等しくなった場合は，4 点のうちの 3 点が一直線上に並んでいることを示している．この後の座標補正を考えると，0 になった場合は交差していると判定したほうがよい．

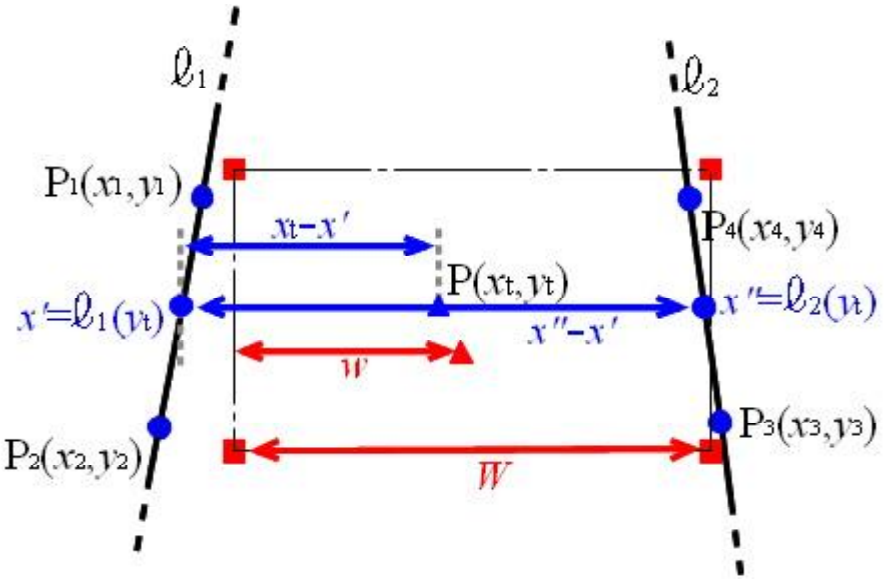

図 22　画面分割による座標補正

図 22 を用いて座標補正方法を説明する．まず，画面をいくつかの四角形の領域に分割する．16 分割，場合によっては 25 分割して，各四角形の頂点をユーザーにタッチしてもらう．16 分割の場合，点の数は 25 個になる．この図ではその中の 4 点に着目する．ユーザーがタッチした点が赤い四角の点であるとし，そのとき得られた電圧の値が示す画面上の点を P_1 〜 P_4 とする．これらの座標は，EEPROM などに保存しておく．ひとつの四角形の頂点から 3 つを選んで先ほどの包含判定を順に行えば，ユーザーのタッチによって検出された座標 $P(x_t,y_t)$がどの四角形に含まれるかが判定できる．その後，P_1 と P_2，P_3 と P_4 を結ぶ直線ℓ_1，ℓ_2 の直線方程式を求める．直線の方程式であるから，x が決まれば y も決まる．両直線の方程式に y_t を代入すれば，直線 $y=y_t$ と両直線の交点の x 座標 x'，x''を求めることができる．ここで，この四角形内の抵抗値分布の歪みがある程度線形に近似できるとすれば，図に示した W(分割した四角形の 1 辺長)と w(四角形の左の辺と，ユーザーのタッチ点との距離)の比は，$x''-x'$と x_t-x'の比にほぼ等しいことがわかる．すなわち，

$$w = \frac{x_t - x'}{x'' - x'} W \tag{12}$$

と見なせる．注目している四角形が左から何番目かを計算して w に適切なオフセットを与えれば，ユーザーがタッチした点の画面上の座標を求めることが出来る．こちらの方法は，ある程度の歪みを修正できるという効果が期待できるが，計算量が多いため，プログラムを作成する際はデータ型や計算手順などに注意する必要がある．この方法を用いても歪みが補正出来ない場合は，そのタッチパネルが壊れていると判断するのが妥当である思われる．

9　活用

前述のように，この機器には多くの改善点が見られる．これらを全て改善できれば，様々に活用できると考えられる．

例えば，近年市場が拡大している電子書籍のブラウザとしての利用が挙げられる．一般的な電子書籍データの閲覧には，対応するソフトが必要な場合が多いが，自分が持っている本をスキャナで画像として取り込めれば，自前の電子書籍として読むことが出来る．特に有用と考えられるのは，広げるのが億劫になるような大きな地図や，重すぎて持ち運べないような図鑑などである．

もうひとつの活用としては，特別何かの機器として扱うのではなく，タッチパネルやUSB メモリの制御の参考としてもらうことである．PIC マイコンを用いて装置を製作するにあたり，タッチパネルで HID を構成したい場合や，巨大な記憶装置として USB メモリを使用したいときの例として利用して頂きたい．

10　謝辞

本装置の製作にあたり，数え切れないほどの文献，資料，ホームページを参考にさせて頂いた．また，指導教官である金野茂男教授，同研究室のメンバー，同学科の学生には多くの協力やアドバイスを賜った．ここに感謝の意を表する．

参考文献

[1] Microchip Technology Inc.,『PIC24FJ256GB110 Family Data Sheet』, 2009 年 3 月

[2] Microchip Technology Inc.,『Implementing File I/O Functions Using Microchip's Disk Drive File System Library』, 2007 年 10 月

[3] ShenZhen Shuang Xiang LCD Tech. Co., Ltd.,『Active matrix 2.4" colour Digital TFT LCD module Technical Product Specification』, 2009 年 5 月

[4] ILI Technology Corp,『ILI9320 - a-Si TFT LCD Single Chip Driver 240RGBx320 Resolution and 16.7M color Datasheet』, 2006 年 4 月

[5] 本澤　上,『PIC24F による USB メモリの読み書き』, 2010 年 9 月

[6]　本澤　上,『PIC マイコンによるタッチパネル付き TFT 液晶の制御』, 2010 年 10 月

[7] nPYO,『妥協の電子工作:4 線抵抗膜方式タッチパネル付き LCD パネルの試用』, http://npyo.web.fc2.com/pages/touch/touch.html, 2010 年 8 月

[8] ChaN,『FatFS モジュール・アプリケーション・ノート』, http://elm-chan.org/fsw/ff/ja/appnote.html, 2011 年 1 月

[9] 文化庁,『内閣告示第 2 号 常用漢字表』, 2010 年 11 月

[5], [6]は本研究室ホームページ(http://www.oyama-ct.ac.jp/D/kinnoken/index.html)からダウンロード可能

付録

プログラムソース - main.c

```
/*********************************************************************
 Project:        TFT_LCD + USB_Memory + Touch_Panel
 FileName:       main.c
 Processor:      PIC24FJ128GB106
 Clock:          32 MHz
 Cycle:          62.5 ns
 Complier:       Microchip C30
 Author:         Joe Honzawa, Oyama National Collage of Technology
*********************************************************************/

//** Includes *****************************************************
#include "usb.h"
#include "usb_host_msd.h"
#include "usb_host_msd_scsi.h"
#include "FSIO.h"
#include "timer.h"
#include "ascii.h"
//*****************************************************************

//** Configuration Bits ********************************************
_CONFIG1(WDTPS_PS1 & FWPSA_PR32 & WINDIS_OFF & FWDTEN_OFF & ICS_PGx1 & GWRP_OFF
& GCP_OFF & JTAGEN_OFF)
_CONFIG2(POSCMOD_HS & IOL1WAY_OFF & OSCIOFNC_ON & FCKSM_CSDCMD & FNOSC_PRIPLL
& PLL_96MHZ_ON & PLLDIV_DIV5 & IESO_ON)
_CONFIG3(WPFP_WPFP0 & WPDIS_WPDIS & WPCFG_WPCFGDIS & WPEND_WPENDMEM)
//*****************************************************************

//** Defines ******************************************************
#define     RED       LATB
#define     GREEN     LATE
#define     BLUE      LATD

#define     HIGH      LATB
#define     MIDDLE    LATE
#define     LOW       LATD
```

```
#define     RS          _LATG9
#define     RD          _LATG8
#define     WR          _LATG7
#define     RESET       _LATG6

#define YM              _LATB12
#define XM              _LATB13
#define YP              _LATB14
#define XP              _LATB15
#define TRIS_YM         _TRISB12
#define TRIS_XM         _TRISB13
#define TRIS_YP         _TRISB14
#define TRIS_XP         _TRISB15

typedef struct PointTag{
        DWORD x;
        DWORD y;
} POINT;
//*********************************************************************

//** Global Variables ************************************************
FSFILE *bmp;
SearchRec rec;
BYTE buf[725];
volatile BOOL deviceAttached;
DWORD Xmax,Ymax,Xmin,Ymin;

struct bmpFiletag{                          // 以下 bmp 用の構造体
        WORD    *Type;                      // 用意したが使用していない
        DWORD   *Size;
        WORD    *Reserved1;
        WORD    *Reserved2;
        DWORD   *OffBits;
}bf;
struct bmpCoretag{
        DWORD   *Size;
        SHORT   *Width;
        SHORT   *Height;
        WORD    *Planes;
```

```
    WORD    *BitCount;
}bc;
struct bmpInfotag{
    DWORD   *Size;
    LONG    *Width;
    LONG    *Height;
    WORD    *Planes;
    WORD    *BitCount;
    DWORD   *Compression;
    DWORD   *SizeImage;
    LONG    *XPixPerMeter;
    LONG    *YPixPerMeter;
    DWORD   *ClrUsed;
    DWORD   *ClrImportant;
}bi;
//*********************************************************************

//** Prototypes ************************************************
void LCD_Initialize(void);
void LCD_Draw_BMP(void);
void LCD_Button(char *str);
void LCD_String(char *str, WORD num, WORD n, WORD m, BYTE r, BYTE g, BYTE b);
void LCD_Char(WORD ch, WORD n, WORD m, BYTE r, BYTE g, BYTE b);
void LCD_BMP_Mode(void);
void LCD_Letter_Mode(void);
void LCD_H_Addr(DWORD x);
void LCD_V_Addr(DWORD y);
void LCD_Cmd_Index(DWORD data);
void LCD_Cmd_Data(DWORD data);
void LCD_Output(DWORD data);
void LCD_GRAM(BYTE red, BYTE green, BYTE blue);
void LCD_GRAM_CLK(void);

void TP_Calibration(void);
POINT TP_Wait_Touch(void);
DWORD TP_Get_X(void);
DWORD TP_Get_Y(void);
CHAR Intersect(POINT p1, POINT p2, POINT p3, POINT p4);
WORD Contain(POINT t1, POINT t2, POINT t3, POINT p);
```

```
void PIC_Initialize(void);
void AD_Initialize(void);
DWORD AD_Convert(WORD n);
//*********************************************************************

//** LCD Functions ***************************************************
void LCD_Initialize(void){
	DWORD i,j;

	RESET=1;
	DelayMs(100);
	RESET=0;
	DelayMs(50);
	RESET=1;
	DelayMs(100);
	RD = 1;

	LCD_Cmd_Index(0x00E3); LCD_Cmd_Data(0x3008); // Set internal timing
	LCD_Cmd_Index(0x00E7); LCD_Cmd_Data(0x0012); // Set internal timing
	LCD_Cmd_Index(0x00EF); LCD_Cmd_Data(0x1231); // Set internal timing
	LCD_Cmd_Index(0x0001); LCD_Cmd_Data(0x0100); // set SS and SM bit
	LCD_Cmd_Index(0x0002); LCD_Cmd_Data(0x0700); // set 1 line inversion
	LCD_Cmd_Index(0x0003); LCD_Cmd_Data(0x1010); // set GRAM write direction and BGR=1.
	LCD_Cmd_Index(0x0004); LCD_Cmd_Data(0x0000); // Resize register
	LCD_Cmd_Index(0x0008); LCD_Cmd_Data(0x0202); // set the back porch and front porch
	LCD_Cmd_Index(0x0009); LCD_Cmd_Data(0x0000); // set non-display area refresh cycle ISC[3:0]
	LCD_Cmd_Index(0x000A); LCD_Cmd_Data(0x0000); // FMARK function
	LCD_Cmd_Index(0x000C); LCD_Cmd_Data(0x0000); // RGB interface setting
	LCD_Cmd_Index(0x000D); LCD_Cmd_Data(0x0000); // Frame marker Position
	LCD_Cmd_Index(0x000F); LCD_Cmd_Data(0x0000); // RGB interface polarity
	//----------Power On sequence ----------//
	LCD_Cmd_Index(0x0010); LCD_Cmd_Data(0x0000);
	LCD_Cmd_Index(0x0011); LCD_Cmd_Data(0x0007);
	LCD_Cmd_Index(0x0012); LCD_Cmd_Data(0x0000); // VREG1OUT voltage
	LCD_Cmd_Index(0x0013); LCD_Cmd_Data(0x0000); // VDV[4:0] for VCOM amplitude
	DelayMs(200); // Dis-charge capacitor power voltage
	LCD_Cmd_Index(0x0010); LCD_Cmd_Data(0x1490);
	LCD_Cmd_Index(0x0011); LCD_Cmd_Data(0x0227);
	DelayMs(50); // DelayMs 50ms
	LCD_Cmd_Index(0x0012); LCD_Cmd_Data(0x001c); // External reference voltage= Vci;
```

```
DelayMs(50); // DelayMs 50ms
LCD_Cmd_Index(0x0013); LCD_Cmd_Data(0x0A00);
LCD_Cmd_Index(0x0029); LCD_Cmd_Data(0x000F);
LCD_Cmd_Index(0x002B); LCD_Cmd_Data(0x000D); // Frame Rate = 91Hz
DelayMs(50); // DelayMs 50ms
LCD_Cmd_Index(0x0020); LCD_Cmd_Data(0x0000); // GRAM horizontal Address
LCD_Cmd_Index(0x0021); LCD_Cmd_Data(0x0000); // GRAM Vertical Address
// ----------- Adjust the Gamma Curve ----------//
LCD_Cmd_Index(0x0030); LCD_Cmd_Data(0x0000);
LCD_Cmd_Index(0x0031); LCD_Cmd_Data(0x0203);
LCD_Cmd_Index(0x0032); LCD_Cmd_Data(0x0001);
LCD_Cmd_Index(0x0035); LCD_Cmd_Data(0x0205);
LCD_Cmd_Index(0x0036); LCD_Cmd_Data(0x030C);
LCD_Cmd_Index(0x0037); LCD_Cmd_Data(0x0607);
LCD_Cmd_Index(0x0038); LCD_Cmd_Data(0x0405);
LCD_Cmd_Index(0x0039); LCD_Cmd_Data(0x0707);
LCD_Cmd_Index(0x003C); LCD_Cmd_Data(0x0502);
LCD_Cmd_Index(0x003D); LCD_Cmd_Data(0x1008);
//------------------ Set GRAM area ----------------//
LCD_Cmd_Index(0x0050); LCD_Cmd_Data(0x0000); // Horizontal GRAM Start Address
LCD_Cmd_Index(0x0051); LCD_Cmd_Data(0x00EF); // Horizontal GRAM End Address
LCD_Cmd_Index(0x0052); LCD_Cmd_Data(0x0000); // Vertical GRAM Start Address
LCD_Cmd_Index(0x0053); LCD_Cmd_Data(0x013F); // Vertical GRAM Start Address
LCD_Cmd_Index(0x0060); LCD_Cmd_Data(0xA700); // Gate Scan Line
LCD_Cmd_Index(0x0061); LCD_Cmd_Data(0x0001); // NDL,VLE); LCD_Cmd_Data(REV
LCD_Cmd_Index(0x006A); LCD_Cmd_Data(0x0000); // set scrolling line
//---------- Partial Display Control ---------//
LCD_Cmd_Index(0x0080); LCD_Cmd_Data(0x0000);
LCD_Cmd_Index(0x0081); LCD_Cmd_Data(0x0000);
LCD_Cmd_Index(0x0082); LCD_Cmd_Data(0x0000);
LCD_Cmd_Index(0x0083); LCD_Cmd_Data(0x0000);
LCD_Cmd_Index(0x0084); LCD_Cmd_Data(0x0000);
LCD_Cmd_Index(0x0085); LCD_Cmd_Data(0x0000);
//---------- Panel Control -------------------//
LCD_Cmd_Index(0x0090); LCD_Cmd_Data(0x0010);
LCD_Cmd_Index(0x0092); LCD_Cmd_Data(0x0600);//0x0000
LCD_Cmd_Index(0x0093); LCD_Cmd_Data(0x0003);
LCD_Cmd_Index(0x0095); LCD_Cmd_Data(0x0110);
LCD_Cmd_Index(0x0097); LCD_Cmd_Data(0x0000);
LCD_Cmd_Index(0x0098); LCD_Cmd_Data(0x0000);
```

```
	LCD_Cmd_Index(0x0007); LCD_Cmd_Data(0x0133); // 262K color and display ON

	LCD_Cmd_Index(0x0022);

	for(i=0 ; i<320 ; i++){
		for(j=0 ; j<240 ; j++){
			LCD_GRAM(0x00, 0x00, 0x00);
		}
	}
}

void LCD_Draw_BMP(void){
	DWORD *offset,val;
	DWORD i,j;

	FSfread(buf, 1, 20, bmp);
	offset = &buf[10];
	FSfseek(bmp, *offset, SEEK_SET);

	LCD_H_Addr(0);
	LCD_V_Addr(319);
	LCD_BMP_Mode();
	for(i=0 ; i<320 ; i++){
		val = FSfread(buf, 1, 720, bmp);
		for(j=0 ; j<val ; j+=3){
			LCD_GRAM(buf[j+2], buf[j+1], buf[j]);
		}
	}
	FSfclose(bmp);
	LCD_Letter_Mode();
}

void LCD_Button(char *str){
	DWORD i,j;

	for(i=0 ; i<21 ; i++){
		LCD_H_Addr(94);
		LCD_V_Addr(284+i);
		for(j=0 ; j<70 ; j++){
			LCD_GRAM(0,0,0);
```

```
            }
        }

        LCD_String(str,4,6,18,255,255,255);
        LCD_H_Addr(93);
        LCD_V_Addr(283);
        for(i=0 ; i<70 ; i++){
            LCD_GRAM(255,255,100);
        }
        LCD_H_Addr(93);
        LCD_V_Addr(305);
        for(i=0 ; i<70 ; i++){
            LCD_GRAM(255,255,100);
        }
        for(i=0 ; i<23 ; i++){
            LCD_V_Addr(283+i);
            LCD_H_Addr(93);
            LCD_GRAM(255,255,100);
            LCD_H_Addr(163);
            LCD_GRAM(255,255,100);
        }
}

void LCD_String(char *str, WORD num, WORD n, WORD m, BYTE r, BYTE g, BYTE b){
        WORD i=0;

        while(*str && i<num){
            LCD_Char(*str++, n++, m, r, g, b);
            if(n >= 15){
                n = 0;
                m++;
                if(m >= 20){
                    m = 0;
                }
            }
            i++;
        }
}

void LCD_Char(WORD ch, WORD n, WORD m, BYTE r, BYTE g, BYTE b){
```

```
	DWORD i,j,x_start,y_start,mask;

	x_start = 16*n;
	y_start = 16*m;
	ch = ch-0x20;

	LCD_H_Addr(x_start);
	LCD_V_Addr(y_start);

	for(i=0 ; i<16 ; i++){
		mask = 0x8000;
		for(j=0 ; j<16 ; j++){
			if(Font[ch][i]&mask){
				LCD_GRAM(r, g, b);
			}else{
				LCD_GRAM(0,0,0);
			}
			mask >>= 1;
		}
		LCD_H_Addr(x_start);
		LCD_V_Addr(y_start+i);
	}
}

void LCD_BMP_Mode(void){
	LCD_Cmd_Index(0x0003);
	LCD_Cmd_Data(0x1010);   // AM=0, I/D=01

	LCD_Cmd_Index(0x0022);
}

void LCD_Letter_Mode(void){
	LCD_Cmd_Index(0x0003);
	LCD_Cmd_Data(0x1030);   // AM=0, I/D=11

	LCD_Cmd_Index(0x0022);
}

void LCD_H_Addr(DWORD x){
	LCD_Cmd_Index(0x0020);
```

```
	LCD_Cmd_Data(x);

	LCD_Cmd_Index(0x0022);
}

void LCD_V_Addr(DWORD y){
	LCD_Cmd_Index(0x0021);
	LCD_Cmd_Data(y);

	LCD_Cmd_Index(0x0022);
}

void LCD_Cmd_Index(DWORD data){
	RS = 0;
	WR = 1;
	LCD_Output(data);
	WR = 0;
	Nop();
	WR = 1;
}

void LCD_Cmd_Data(DWORD data){
	RS = 1;
	WR = 1;
	LCD_Output(data);
	WR = 0;
	Nop();
	WR = 1;
}

void LCD_Output(DWORD data){
	DWORD data_H, data_M, data_L;

	data_H = data>>8;
	data_M = data>>3;
	data_L = data<<3;

	HIGH = (HIGH&0xFF00) | (data_H&0x00FF);
	MIDDLE = data_M;
	LOW = (LOW&0xFF00) | (data_L&0x00FF);
```

```
}

void LCD_GRAM(BYTE red, BYTE green, BYTE blue){
	RS = 1;
	WR = 1;
	RED = (RED&0xFF00) | red;
	GREEN = green;
	BLUE = (BLUE&0xFF00) | blue;
	WR = 0;
	Nop();
	WR = 1;
}

void LCD_GRAM_CLK(void){
	RS = 1;
	WR = 1;
	Nop();
	WR = 0;
	Nop();
	WR = 1;
}
//*********************************************************************

//** Touch Panel Functions ****************************************
void TP_Calibration(void){
	DWORD x80,y100,x200,y250;

	LCD_H_Addr(80);
	LCD_V_Addr(100);
	LCD_GRAM(255,255,255);
	while(TP_Get_X() < 0x0020);
	DelayMs(300);
	x80 = TP_Get_X();
	y100 = TP_Get_Y();

	while(TP_Get_X() > 0x0020);
	DelayMs(300);

	LCD_H_Addr(200);
	LCD_V_Addr(250);
```

```
	LCD_GRAM(255,255,255);
	while(TP_Get_X() < 0x0020);
	DelayMs(300);
	x200 = TP_Get_X();
	y250 = TP_Get_Y();

	Xmin = x80 - (x200-x80)*2/3;
	Ymin = y100 - (y250-y100)*2/3;
	Xmax = x200 + (x200-x80)*39/120;
	Ymax = y250 + (y250-y100)*69/150;
}

POINT TP_Wait_Touch(void){
	WORD i;
	POINT p[2];

	while(1){
		while(TP_Get_X()<0x0020);
		for(i=0 ; i<2 ; i++){
			p[i].x = TP_Get_X();
			p[i].y = TP_Get_Y();
		}
		if(p[0].x==p[1].x && p[0].y==p[1].y){
			p[0].x = 239*(p[1].x-Xmin)/(Xmax-Xmin);
			p[0].y = 319*(p[1].y-Ymin)/(Ymax-Ymin);
			return p[0];
		}
	}
}

DWORD TP_Get_X(void){
	DWORD i;
	DWORD sum,res;

	TRISB = 0x5000;
	AD1CON1bits.ADON = 0;
	AD1PCFGL = 0xAFFF;
	AD1CON1bits.ADON = 1;
	XP = 0;
	for(i=0,sum=0 ; i<32 ; i++){
```

```
        XM = 1;
        res = AD_Convert(12);
        res += AD_Convert(14);
        XM = 0;
        res >>= 1;
        sum += res;
    }

    return sum>>5;
}

DWORD TP_Get_Y(void){
    DWORD i;
    DWORD sum,res;

    TRISB = 0xA000;
    AD1CON1bits.ADON = 0;
    AD1PCFGL = 0x5FFF;
    AD1CON1bits.ADON = 1;
    YP = 0;
    for(i=0,sum=0 ; i<32 ; i++){
        YM = 1;
        res = AD_Convert(13);
        res += AD_Convert(15);
        YM = 0;
        res >>= 1;
        sum += res;
    }

    return sum>>5;
}


CHAR Intersect(POINT p1, POINT p2, POINT p3, POINT p4){    //  p1,p2 を結ぶ線分と、p3,p4 を結ぶ線分の交差を判定
    return      (p1.x-p2.x)*(p3.y-p1.y) + ¥
                (p1.y-p2.y)*(p1.x-p3.x)*(p1.x-p2.x)*(p4.y-p1.y) + ¥
                (p1.y-p2.y)*(p1.x-p4.x);                    // >0:交差しない
                                                            // <0:交差する
                                                            // =0:交差する(一直線上に 3 点以上が存在)
```

```
}

WORD Contain(POINT t1, POINT t2, POINT t3, POINT p){        // 三角形(t1,t2,t3)に p が含まれるか判定
      if(Intersect(t1, t2, t3, p) <= 0) return 0;                    // 0:含まれない
      if(Intersect(t2, t3, t1, p) <= 0) return 0;                    // 1:含まれる
      if(Intersect(t3, t1, t2, p) <= 0) return 0;
      return 1;
}
//*********************************************************************

//** PIC Functions ***************************************************
void PIC_Initialize(void){
      TRISB = 0xF000;      LATB = 0x0000;
      TRISC = 0x0000;      LATC = 0x0000;
      TRISD = 0x0000;      LATD = 0x0000;
      TRISE = 0x0000; LATE = 0x0000;
      TRISF = 0x0000; LATF = 0x0000;
      TRISG = 0x0000;      LATG = 0x0000;

     deviceAttached = FALSE;
}

void AD_Initialize(void){
      AD1CON1bits.ADON = 0;        // A/D コンバータ無効 関連レジスタ操作の際は無効にするべき

      AD1PCFGL = 0x0FFF;           // 1:デジタルピン  0:アナログピン

      AD1CON1bits.ADSIDL = 1;      // アイドル中は動作停止

      AD1CON1bits.FORM1 = 0;       // 00:符号無し整数
      AD1CON1bits.FORM0 = 0;       // 11:符号なし少数、10:少数、01:整数

      AD1CON1bits.SSRC2 = 1;
      AD1CON1bits.SSRC1 = 1;       // 111:サンプリング後に自動変換
      AD1CON1bits.SSRC0 = 1;

      AD1CON1bits.ASAM = 0;        // SAMP ビットでサンプリング開始
      AD1CON1bits.SAMP = 0;        // サンプリング停止

      AD1CON2bits.VCFG2 = 0;
```

```
    AD1CON2bits.VCFG1 = 0;      // 000:AVDD,AVSS をリファレンス電圧とする
    AD1CON2bits.VCFG0 = 0;

    AD1CON2bits.CSCNA = 0;      // MUXA でスキャンする

    AD1CON2bits.SMPI3 = 0;
    AD1CON2bits.SMPI2 = 0;      // 0000:変換完了毎に割り込みフラグが立つ
    AD1CON2bits.SMPI1 = 0;
    AD1CON2bits.SMPI0 = 0;

    AD1CON2bits.BUFM = 0;       // バッファは 16 ビット
    AD1CON2bits.ALTS = 0;       // 常に MUXA を使用する

    AD1CON3bits.ADRC = 0;       // クロック源はシステムクロック

    AD1CON3bits.SAMC4 = 1;
    AD1CON3bits.SAMC3 = 1;
    AD1CON3bits.SAMC2 = 1;      // 11100:28Tad
    AD1CON3bits.SAMC1 = 0;
    AD1CON3bits.SAMC0 = 0;

    AD1CON3bits.ADCS7 = 0;
    AD1CON3bits.ADCS6 = 0;
    AD1CON3bits.ADCS5 = 0;
    AD1CON3bits.ADCS4 = 0;      // 00001001:3Tcy を 1Tad とする
    AD1CON3bits.ADCS3 = 1;
    AD1CON3bits.ADCS2 = 0;
    AD1CON3bits.ADCS1 = 0;
    AD1CON3bits.ADCS0 = 1;

    AD1CSSL = 0x0000;           // 0:スキャンしない 1:スキャンする

    AD1CON1bits.ADON = 1;       // A/D コンバータ有効
}

DWORD AD_Convert(WORD n){
    AD1CHS = n & 0x000F;
    AD1CON1bits.SAMP = 1;
    Nop(); Nop(); Nop();
    Nop(); Nop(); Nop();
```

```
	while(!AD1CON1bits.DONE);

	return ADC1BUF0&0x0FFF;
}
//*********************************************************************

//** Main Function ***************************************************
int main(void){
	DWORD nof, nod, selected=25, drew=0, cd_pos, i, j;
	DWORD row, column, dir_row=0;
	char cd[] = ".                ";
	POINT p;

	DelayMs(500);

	USBInitialize(0);

	PIC_Initialize();
	AD_Initialize();
	LCD_Initialize();

	TP_Calibration();
	LCD_H_Addr(0);
	LCD_V_Addr(0);

	RED = (RED&0xFF00);
	GREEN = 0x00;
	BLUE = (BLUE&0xFF00);
	for(i=0 ; i<320 ; i++){
		for(j=0 ; j<240 ; j++){
			LCD_GRAM_CLK();
		}
	}

	while(1){
		USBTasks();

		if(USBHostMSDSCSIMediaDetect()){
			deviceAttached = TRUE;
```

```
if(FSInit()){
    while(1){
        nof = 0;
        nod = 0;
        cd_pos = 0;
        if(!FindFirst("*.bmp", ATTR_MASK, &rec)){
            LCD_String("FILE LIST",9,0,0,150,150,255);
            LCD_String(rec.filename,12,1,2,255,255,255);
            nof = 1;
            while(!FindNext(&rec) && nof<10){
                nof++;
                LCD_String(rec.filename,12,1,nof+1,255,255,255);
            }
        }
        if(!FindFirst("*", ATTR_DIRECTORY, &rec)){
            if(strcmp(rec.filename, cd)){
                cd_pos = 1;
                FindNext(&rec);
            }
            dir_row = (nof==0 ? 0 : nof+3);
            LCD_String("DIR LIST",8,0,dir_row,150,150,255);
            LCD_String(rec.filename,12,1,dir_row+2,255,255,255);
            nod = 1;
            while(!FindNext(&rec) && nod<5){
                if(strcmp(rec.filename, cd) && cd_pos==0){
                    cd_pos = nod+1;
                }else{
                    nod++;
                    LCD_String(rec.filename,12,1,dir_row+nod+1,255,255,255);
                }
            }
        }
        if(nof==0 && nod==0){
            LCD_String("NO FILES.",9,4,10,255,50,50);
        }

        selected = 25;
        while(1){
            p = TP_Wait_Touch();
            row = p.y/16;
```

```
column = p.x/16;
if(nof!=0 && row>=2 && row<=nof+1){
	if(selected != 25){
		LCD_Char(' ',0,selected,0,0,0);
	}
	selected = row;
	LCD_Char('>',0,selected,150,150,255);
	LCD_Button("VIEW");
	drew = 0;
}
if(nod!=0 && row>=dir_row+2 && row<=dir_row+nod+1){
	if(selected != 25){
		LCD_Char(' ',0,selected,0,0,0);
	}
	selected = row;
	LCD_Char('>',0,selected,150,150,255);
	LCD_Button("OPEN");
	drew = 0;
}
if(selected!=25 && row==18 && 6<=column && column<=9){
	if(selected>=2 && selected<=(nof+1)){
		FindFirst("*.bmp", ATTR_MASK, &rec);i=1;
		while(i<selected-1){
			FindNext(&rec);
			i++;
		}
		bmp = FSfopen(rec.filename, READ);
		LCD_Draw_BMP();
		LCD_Button("BACK");
		drew = 1;
		while(1){
			p = TP_Wait_Touch();
			row = p.y/16;
			column = p.x/16;
			if(row==18 && 6<=column && column<=9){
				drew = 1;
				break;
			}
		}
	}
```

```
                        if(dir_row!=0 && selected>=dir_row+2 && selected<=(dir_row+nod+1)){
                                FindFirst("*", ATTR_DIRECTORY, &rec);i=1;
                                if(selected >= dir_row+cd_pos+1){
                                        i--;
                                }
                                while(i<selected-dir_row-1){
                                        FindNext(&rec);
                                        i++;
                                }
                                FSchdir(rec.filename);
                                drew = 1;
                        }
                        if(drew){
                                LCD_H_Addr(0);
                                LCD_V_Addr(0);
                                RED = (RED&0xFF00);
                                GREEN = 0x00;
                                BLUE = (BLUE&0xFF00);
                                for(i=0 ; i<320 ; i++){
                                        for(j=0 ; j<240 ; j++){
                                                LCD_GRAM_CLK();
                                        }
                                }
                                drew = 0;
                                break;
                        }
                    }
                }
                while(deviceAttached == TRUE){              // USB メモリが取り外されるまで待機
                    USBTasks();
                }
            }
        }
    }
    return 0;
}

/*****************************************************************************
```

```
  Function:
    BOOL USB_ApplicationEventHandler( BYTE address, USB_EVENT event,
                        void *data, DWORD size )

  Summary:
    This is the application event handler.  It is called when the stack has
    an event that needs to be handled by the application layer rather than
    by the client driver.

  Description:
    This is the application event handler.  It is called when the stack has
    an event that needs to be handled by the application layer rather than
    by the client driver.  If the application is able to handle the event, it
    returns TRUE.  Otherwise, it returns FALSE.

  Precondition:
    None

  Parameters:
    BYTE address      - Address of device where event occurred
    USB_EVENT event - Identifies the event that occured
    void *data          - Pointer to event-specific data
    DWORD size          - Size of the event-specific data

  Return Values:
    TRUE      - The event was handled
    FALSE     - The event was not handled

  Remarks:
    The application may also implement an event handling routine if it
    requires knowledge of events.  To do so, it must implement a routine that
    matches this function signature and define the USB_HOST_APP_EVENT_HANDLER
    macro as the name of that function.
  ***************************************************************************/

BOOL USB_ApplicationEventHandler( BYTE address, USB_EVENT event, void *data, DWORD size )
{
    switch( event )
    {
        case EVENT_VBUS_REQUEST_POWER:
```

```
        // The data pointer points to a byte that represents the amount of power
        // requested in mA, divided by two.  If the device wants too much power,
        // we reject it.
        return TRUE;

    case EVENT_VBUS_RELEASE_POWER:
        // Turn off Vbus power.
        // The PIC24F with the Explorer 16 cannot turn off Vbus through software.

        //This means that the device was removed
        deviceAttached = FALSE;
        return TRUE;
        break;

    case EVENT_HUB_ATTACH:
        return TRUE;
        break;

    case EVENT_UNSUPPORTED_DEVICE:
        return TRUE;
        break;

    case EVENT_CANNOT_ENUMERATE:
        //UART2PrintString( "¥r¥n***** USB Error - cannot enumerate device *****¥r¥n" );
        return TRUE;
        break;

    case EVENT_CLIENT_INIT_ERROR:
        //UART2PrintString( "¥r¥n***** USB Error - client driver initialization error *****¥r¥n" );
        return TRUE;
        break;

    case EVENT_OUT_OF_MEMORY:
        //UART2PrintString( "¥r¥n***** USB Error - out of heap memory *****¥r¥n" );
        return TRUE;
        break;

    case EVENT_UNSPECIFIED_ERROR:   // This should never be generated.
        //UART2PrintString( "¥r¥n***** USB Error - unspecified *****¥r¥n" );
        return TRUE;
```

```
                break;

            default:
                break;
        }

        return FALSE;
}
```

プログラムソース - ascii.h

```
//*****************************************************************
//      ASCII フォントデータ  16x16 ドット
//*****************************************************************

const DWORD Font[95][16] = {
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000},   /*Space*/
{0x0000,0x0180,0x0180,0x0180,0x0180,0x0180,0x0100,0x0100,0x0100,0x0100,0x0000,0x0000,0x0180,0x0180,0x0000,0x0000},   /* ! */
{0x0036,0x0036,0x0012,0x0024,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000},   /* " */
{0x0210,0x0210,0x0210,0x0210,0x1FFC,0x1FFC,0x0420,0x0420,0x0420,0x0420,0x3FF8,0x3FF8,0x0840,0x0840,0x0840,0x0840}, /* # */
{0x0000,0x0100,0x07E0,0x0930,0x1118,0x1100,0x0900,0x0700,0x01E0,0x0110,0x0108,0x1108,0x1910,0x0FE0,0x0100,0x0000},   /* $ */
{0x0000,0x3802,0x4404,0x4408,0x4410,0x4420,0x4440,0x3880,0x011C,0x0222,0x0422,0x0822,0x1022,0x2022,0x401C,0x0000},   /* % */
{0x0000,0x0F00,0x1880,0x1080,0x1180,0x1B00,0x0E7C,0x0C10,0x1E30,0x3320,0x61E0,0x40C0,0x41E2,0x2336,0x1E1C,0x0000},/* & */
{0x1800,0x1800,0x0800,0x1000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000},   /* ' */
{0x0008,0x0010,0x0030,0x0020,0x0060,0x0040,0x0040,0x0040,0x0040,0x0040,0x0040,0x0060,0x0020,0x0030,0x0010,0x0008},   /* ( */
{0x1000,0x0800,0x0C00,0x0400,0x0600,0x0200,0x0200,0x0200,0x0200,0x0200,0x0200,0x0600,0x0400,0x0C00,0x0800,0x1000},   /* ) */
{0x0000,0x0000,0x0000,0x0000,0x0100,0x1930,0x0D60,0x07C0,0x0100,0x07C0,0x0D60,0x1930,0x0100,0x0000,0x0000,0x0000}, /* * */
{0x0000,0x0000,0x0100,0x0100,0x0100,0x0100,0x0100,0x3FFC,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0000,0x0000},   /* + */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x6000,0x6000,0x2000,0x4000},   /* , */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3FFC,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000},   /* - */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x6000,0x6000,0x0000,0x0000},   /* . */
{0x0001,0x0002,0x0004,0x0008,0x0010,0x0020,0x0040,0x0080,0x0100,0x0200,0x0400,0x0800,0x1000,0x2000,0x4000,0x8000},   /* / */
{0x0000,0x03C0,0x0420,0x0810,0x0810,0x1818,0x1008,0x1008,0X1008,0x1008,0x1008,0x0810,0x0810,0x0420,0x03C0,0x0000}, /* 0 */
{0x0000,0x0100,0x0300,0x0500,0x0900,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x07C0,0x0000},   /* 1 */
{0x0000,0x07E0,0x0C10,0x1808,0x1008,0x0008,0x0018,0x0030,0x0060,0x00C0,0x0180,0x0300,0x0600,0x0C00,0x1FF8,0x0000}, /* 2 */
{0x0000,0x03E0,0x0E10,0x1808,0x0008,0x0008,0x0010,0x07E0,0x0010,0x0008,0x0008,0x0008,0x0018,0x0070,0x1FC0,0x0000}, /* 3 */
{0x0000,0x0060,0x00E0,0x01A0,0x0320,0x0620,0x0C20,0x1820,0x3020,0x2020,0x3FFC,0x0020,0x0020,0x0020,0x00F8,0x0000}, /* 4 */
{0x0000,0x1FF8,0x1000,0x1000,0x1000,0x1000,0x17E0,0x1810,0x1008,0x0008,0x0008,0x0018,0x0030,0x00E0,0x1F80,0x0000},   /* 5 */
{0x0000,0x00F0,0x0380,0x0600,0x0C00,0x0800,0x1BC0,0x1430,0x1818,0x1008,0x1008,0x1808,0x0818,0x0C30,0x03C0,0x0000}, /* 6 */
{0x0000,0x1FFC,0x1008,0x1018,0x1010,0x0030,0x0020,0x0060,0x0040,0x00C0,0x0080,0x0080,0x0180,0x0100,0x0100,0x0000},   /* 7 */
{0x0000,0x03C0,0x0C30,0x1818,0x1008,0x1008,0x0810,0x07E0,0x0810,0x1008,0x1008,0x1008,0x1818,0x0C30,0x03C0,0x0000}, /* 8 */
{0x0000,0x03C0,0x0c30,0x1810,0x1018,0x1008,0x1008,0x1818,0x0C28,0x03D8,0x0010,0x0030,0x0060,0x01C0,0x0F00,0x0000}, /* 9 */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0180,0x0180,0x0000,0x0000,0x0000,0x0000,0x0180,0x0180,0x0000,0x0000},   /* : */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0180,0x0180,0x0000,0x0000,0x0000,0x0000,0x0180,0x0180,0x0080,0x0100},   /* ; */
{0x0000,0x0006,0x0018,0x0060,0x0180,0x0600,0x1800,0x6000,0x1800,0x0400,0x0300,0x00C0,0x0020,0x0018,0x0006,0x0000},   /* < */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x3FFC,0x0000,0x0000,0x0000,0x3FFC,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000}, /* = */
{0x0000,0x6000,0x1800,0x0600,0x0180,0x0060,0x0018,0x0006,0x0018,0x0020,0x00C0,0x0300,0x0400,0x1800,0x6000,0x0000},   /* > */
{0x0000,0x07E0,0x0810,0x1008,0x1008,0x0808,0x0030,0x00C0,0x0100,0x0100,0x0000,0x0000,0x0180,0x0180,0x0000,0x0000},   /* ? */
```

```
{0x0000,0x0000,0x03C0,0x0C30,0x1008,0x11A8,0x2264,0x2444,0x2444,0x24C8,0x1370,0x1004,0x0C18,0x03E0,0x0000,0x0000}, /* @ */
{0x0000,0x0100,0x0380,0x0280,0x06C0,0x0440,0x0C60,0x0820,0x0FE0,0x1830,0x1010,0x3018,0x2008,0xF83E,0x0000,0x0000}, /* A */
{0x0000,0x3FE0,0x0810,0x0808,0x0808,0x0808,0x0810,0x0FE0,0x0818,0x0804,0x0804,0x0804,0x0808,0x3FF0,0x0000,0x0000}, /* B */
{0x0000,0x03C8,0x0C28,0x1818,0x1008,0x3008,0x2000,0x2000,0x2000,0x3000,0x1008,0x1818,0x0C30,0x03C0,0x0000,0x0000}, /* C */
{0x0000,0x3FC0,0x0830,0x0818,0x0808,0x080C,0x0804,0x0804,0x0804,0x080C,0x0808,0x0818,0x0830,0x3FC0,0x0000,0x0000}, /* D */
{0x0000,0x3FF8,0x0808,0x0804,0x0800,0x0840,0x0840,0x0FC0,0x0840,0x0840,0x0800,0x0804,0x0808,0x3FF8,0x0000,0x0000}, /* E */
{0x0000,0x1FFC,0x0404,0x0404,0x0400,0x0420,0x0420,0x07E0,0x0420,0x0420,0x0400,0x0400,0x0400,0x1F00,0x0000,0x0000}, /* F */
{0x0000,0x03C8,0x0C38,0x1808,0x1008,0x3000,0x2000,0x2000,0x203E,0x3008,0x1008,0x1818,0x0C28,0x03C8,0x0000,0x0000}, /* G */
{0x0000,0x7C3E,0x1008,0x1008,0x1008,0x1008,0x1008,0x1FF8,0x1008,0x1008,0x1008,0x1008,0x1008,0x7C3E,0x0000,0x0000}, /* H */
{0x0000,0x07C0,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x07C0,0x0000,0x0000}, /* I */
{0x0000,0x00F8,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x1020,0x1840,0x0F80,0x0000,0x0000}, /* J */
{0x0000,0x3E7C,0x0830,0x0860,0x08C0,0x0980,0x0B00,0x0F00,0x0980,0x08C0,0x0860,0x0830,0x0818,0x3E3E,0x0000,0x0000}, /* K */
{0x0000,0x1F00,0x0400,0x0400,0x0400,0x0400,0x0400,0x0400,0x0400,0x0400,0x0404,0x0404,0x0404,0x1FFC,0x0000,0x0000}, /* L */
{0x0000,0xF01E,0x3018,0x3838,0x2828,0x2828,0x2C68,0x2448,0x2448,0x26C8,0x2288,0x2388,0x2108,0xF11E,0x0000,0x0000}, /* M */
{0x0000,0x783E,0x1C08,0x1408,0x1608,0x1308,0x1108,0x1188,0x10C8,0x1048,0x1068,0x1038,0x1018,0x7808,0x0000,0x0000}, /* N */
{0x0000,0x03C0,0x0C30,0x1818,0x1008,0x300C,0x2004,0x2004,0x2004,0x300C,0x1008,0x1818,0x0C30,0x03C0,0x0000,0x0000}, /* O */
{0x0000,0x1FF0,0x0408,0x0404,0x0404,0x0404,0x0408,0x07F0,0x0400,0x0400,0x0400,0x0400,0x0400,0x1F00,0x0000,0x0000}, /* P */
{0x0000,0x03C0,0x0C30,0x1818,0x1008,0x300C,0x2004,0x2004,0x2004,0x300C,0x1188,0x1A58,0x0C30,0x03D2,0x000C,0x0000}, /* Q */
{0x0000,0x3FE0,0x0810,0x0808,0x0808,0x0808,0x0810,0x0FE0,0x0820,0x0830,0x0810,0x0818,0x0808,0x3E3E,0x0000,0x0000}, /* R */
{0x0000,0x07C8,0x0828,0x1018,0x1008,0x1008,0x0C00,0x03C0,0x0030,0x1008,0x1008,0x1808,0x1410,0x13E0,0x0000,0x0000}, /* S */
{0x0000,0x7FFC,0x4104,0x4104,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x07C0,0x0000,0x0000}, /* T */
{0x0000,0x7C3E,0x1008,0x1008,0x1008,0x1008,0x1008,0x1008,0x1008,0x1008,0x1008,0x1818,0x0C30,0x03C0,0x0000,0x0000}, /* U */
{0x0000,0x7C3E,0x1008,0x1818,0x0810,0x0830,0x0C20,0x0420,0x0460,0x0640,0x02C0,0x0280,0x0380,0x0100,0x0000,0x0000}, /* V */
{0x0000,0xF99F,0x2148,0x2148,0x23C4,0x324C,0x1248,0x1668,0x1428,0x1428,0x1C38,0x0C30,0x0810,0x0810,0x0000,0x0000}, /* W */
{0x0000,0x7C7C,0x1010,0x1830,0x0C60,0x06C0,0x0380,0x0100,0x0380,0x06C0,0x0C60,0x1830,0x1010,0x787C,0x0000,0x0000}, /* X */
{0x0000,0x7C7C,0x1010,0x1830,0x0820,0x0C60,0x0440,0x06C0,0x0380,0x0100,0x0100,0x0100,0x0100,0x07C0,0x0000,0x0000}, /* Y */
{0x0000,0x1FF8,0x1030,0x1060,0x0040,0x00C0,0x0180,0x0100,0x0300,0x0600,0x0400,0x0C08,0x1808,0x3FF8,0x0000,0x0000}, /* Z */
{0x00f8,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x00f8}, /* [ */
{0x0000,0x7c3e,0x1008,0x1008,0x0810,0x7ffe,0x0420,0x0440,0x0280,0x7ffe,0x0100,0x0100,0x0100,0x0100,0x07c0,0x0000}, /*¥ */
{0x1f00,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x1f00}, /* ] */
{0x0180,0x0240,0x0420,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000}, /* ^ */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xffff}, /* _ */
{0x0300,0x0100,0x0080,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000}, /* ` */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07c0,0x0020,0x0020,0x07e0,0x0820,0x0820,0x0868,0x0798,0x0000,0x0000}, /* a */
{0x0000,0x1c00,0x0400,0x0400,0x0400,0x0400,0x05e0,0x0610,0x0408,0x0408,0x0408,0x0408,0x0610,0x05e0,0x0000,0x0000}, /* b */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03e0,0x0410,0x0800,0x0800,0x0800,0x0800,0x0410,0x03e0,0x0000,0x0000}, /* c */
{0x0000,0x00e0,0x0020,0x0020,0x0020,0x0020,0x07a0,0x0860,0x1020,0x1020,0x1020,0x1020,0x0860,0x07b8,0x0000,0x0000}, /* d */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03c0,0x0420,0x0810,0x0ff0,0x0800,0x0800,0x0410,0x03e0,0x0000,0x0000}, /* e */
{0x0000,0x0078,0x0088,0x0100,0x0100,0x0100,0x0fe0,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x07c0,0x0000,0x0000}, /* f */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07d8,0x0820,0x0820,0x0820,0x07c0,0x0800,0x0ff0,0x1008,0x1008,0x0ff0}, /* g */
```

```
{0x0000,0x1c00,0x0400,0x0400,0x0400,0x0400,0x05e0,0x0630,0x0410,0x0410,0x0410,0x0410,0x0410,0x1e3c,0x0000,0x0000},   /* h */
{0x0000,0x0000,0x0180,0x0180,0x0000,0x0000,0x0380,0x0080,0x0080,0x0080,0x0080,0x0080,0x0080,0x03e0,0x0000,0x0000},   /* i */
{0x0000,0x0000,0x00c0,0x00c0,0x0000,0x0000,0x01c0,0x0040,0x0040,0x0040,0x0040,0x0040,0x0040,0x1080,0x1f00},   /* j */
{0x0000,0x3800,0x0800,0x0800,0x0800,0x0800,0x08f8,0x0860,0x0980,0x0e00,0x0900,0x08c0,0x0830,0x3e7c,0x0000,0x0000},   /* k */
{0x0000,0x0700,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x07c0,0x0000,0x0000},   /* l */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3738,0x18c4,0x1084,0x1084,0x1084,0x1084,0x1084,0x39ce,0x0000,0x0000},   /* m */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x1de0,0x0630,0x0410,0x0410,0x0410,0x0410,0x0410,0x1e3c,0x0000,0x0000},   /* n */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03c0,0x0420,0x0810,0x0810,0x0810,0x0810,0x0420,0x03c0,0x0000,0x0000},   /* o */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x1de0,0x0610,0x0408,0x0408,0x0408,0x0610,0x05e0,0x0400,0x0400,0x1f00},   /* p */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07a0,0x0860,0x1020,0x1020,0x1020,0x0860,0x07a0,0x0020,0x0020,0x00f8},   /* q */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0e70,0x0290,0x0300,0x0200,0x0200,0x0200,0x0200,0x0f80,0x0000,0x0000},   /* r */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07d0,0x0830,0x0810,0x0780,0x0060,0x0810,0x0c10,0x0be0,0x0000,0x0000},   /* s */
{0x0000,0x0000,0x0200,0x0200,0x0200,0x0200,0x1fe0,0x0200,0x0200,0x0200,0x0200,0x0200,0x0110,0x00f0,0x0000,0x0000},   /* t */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x1c70,0x0410,0x0410,0x0410,0x0410,0x0410,0x0630,0x03dc,0x0000,0x0000},   /* u */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3c78,0x0820,0x0c60,0x0440,0x06c0,0x0280,0x0380,0x0100,0x0000,0x0000},   /* v */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x799e,0x1188,0x1bd8,0x0a50,0x0a50,0x0e70,0x0420,0x0420,0x0000,0x0000},   /* w */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x1e78,0x0420,0x0240,0x0180,0x0240,0x0420,0x0810,0x3e7c,0x0000,0x0000},   /* x */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x1e3c,0x0410,0x0630,0x0220,0x0360,0x0140,0x00c0,0x0180,0x1300,0x1e00},   /* y */
{0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0ff0,0x0820,0x0840,0x0080,0x0100,0x0210,0x0410,0x0ff0,0x0000,0x0000},   /* z */
{0x0008,0x0010,0x0020,0x0020,0x0020,0x0020,0x0020,0x0040,0x0020,0x0020,0x0020,0x0020,0x0020,0x0020,0x0010,0x0008},   /* { */
{0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100,0x0100},   /* | */
{0x1000,0x0800,0x0400,0x0400,0x0400,0x0400,0x0400,0x0200,0x0400,0x0400,0x0400,0x0400,0x0400,0x0400,0x0800,0x1000},   /* } */
{0x0000,0x0710,0x08e0,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000}    /* ~ */
};
```

パターン図の貼り方

本書中の図 6 のように，ワープロソフトの文書中に解像度の高いパターン図を貼り付けたいときがある．その手順を説明する．使用するソフトウェアを以下に示す．

・PCBE：プリント基板パターンエディタ[*1]
・PDFCreater：PDF 作成ソフト
・PDF-XChange Viewer：フリーの PDF ファイルビューワー

はじめに，PCBE を使ってパターン図を作成する．著者の場合は部品面から見たパターンを作成して反転印刷を行っているが，各自やり易いように作成して頂きたい．パターン図が出来たら，PDFCreater をプリンタとして指定し，印刷を行う．このとき注意するべきことは，倍率は 1 のままにしておかなければならないことである．この値を変更すると，最終的に貼り付ける際に面倒になる．また，PDFCreater の設定で，印刷品質をある程度(300dpi から 600dpi)に上げておいたほうがよい．

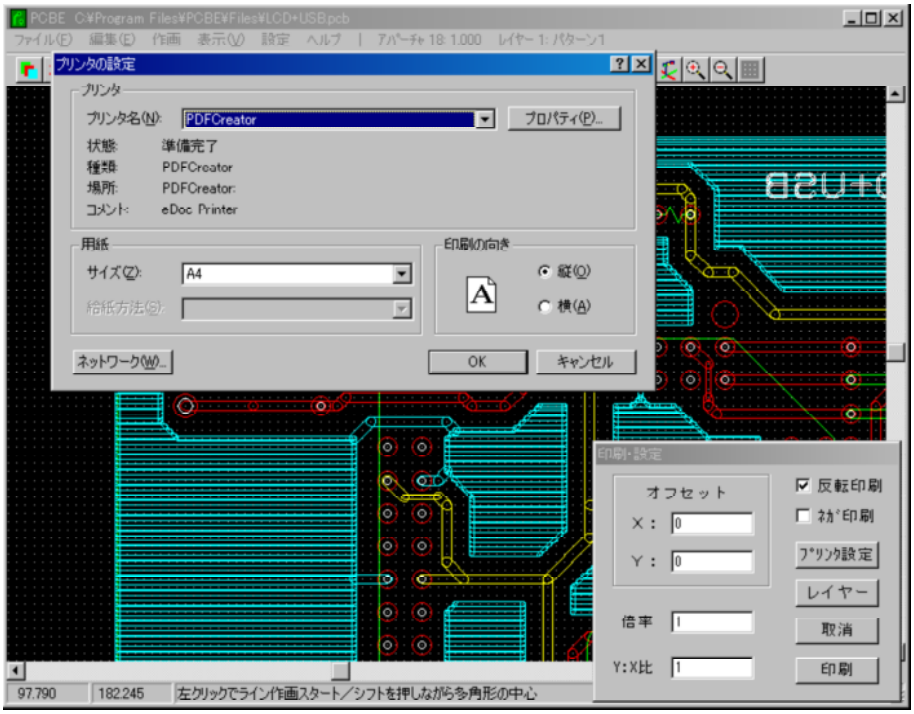

図 23　PCBE からの PDF 作成

*1 大規模なバージョンアップが行われ，v0.50 以降では旧バージョン以前のファイルが読み込めない場合があるため注意が必要である．

次に，印刷された PDF ファイルを PDF-XChange Viewer で開く．「ファイル」→「エクスポート」→「イメージへエクスポート」を選択し，ズームを 100%，解像度を 300dpi から 600dpi 程度に設定する．保存先のフォルダを確認してエクスポートする．画像のフォーマットは，BMP が望ましい．以上で貼り付ける画像を作成することが出来た．あとはワープロソフトから文中に画像を貼り付け，一太郎であれば画像右クリックから「等倍に戻す」，Word であれば画像のサイズを 100%とすればよい．パターン図の周りの余分な空白は，予めペイントソフトで切り取っておくか，ワープロソフトでトリミングすればよい．

プリント基板の製作テクニック

ここでは，本研究室で行っているプリント基板製作の方法について述べる．詳しい手順は多くの文献が解説しているので，知っておくとプリント基板製作がより手軽なものとなるようなテクニックを紹介する．

①パターン図の印刷

パターン図をフィルムに印刷する際，一般的に使用される専用のフィルムは非常に高価である．出来上がりは綺麗になると思われるが，失敗してしまったときを考えるとなかなか手が出ない．そこで本研究室では，OHP フィルムを使用している．近頃はプレゼンテーションに OHP を使用することは稀であるが，いまだに 1 枚 20 円から 30 円程度で手に入る．このフィルムに，レーザープリンタで同一のパターン図を 4 つ印刷し，重ねて貼ることで，ある程度専用フィルムの代用が出来る．ただし，同時に 4 つ印刷したとしても若干のずれが生じてしまうため，丁度よく重なることはあまり無い．非常に細かいパターンを印刷する際は注意する必要がある．

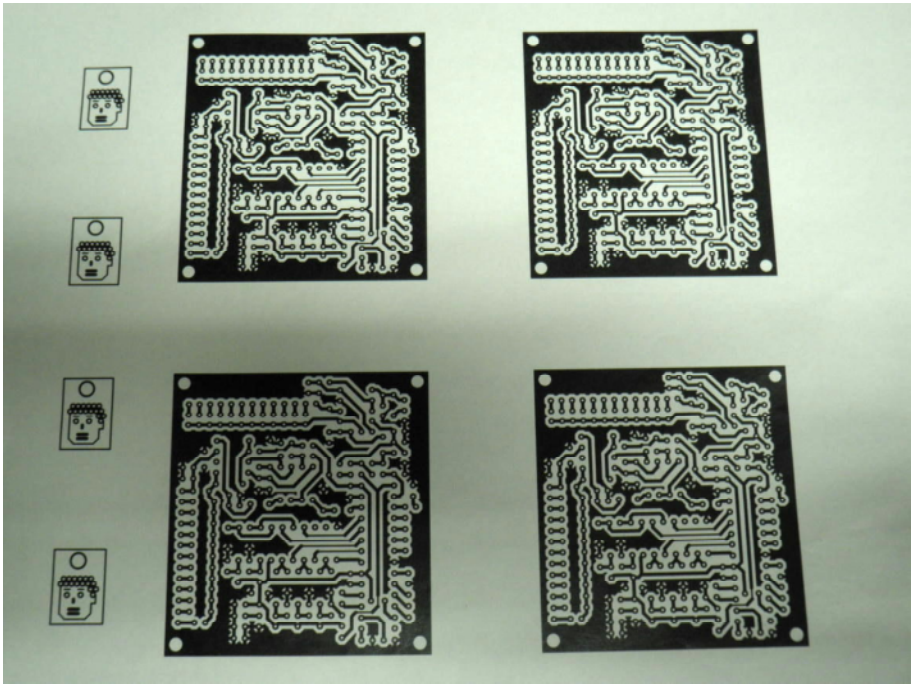

図 24　普通紙に印刷したパターン

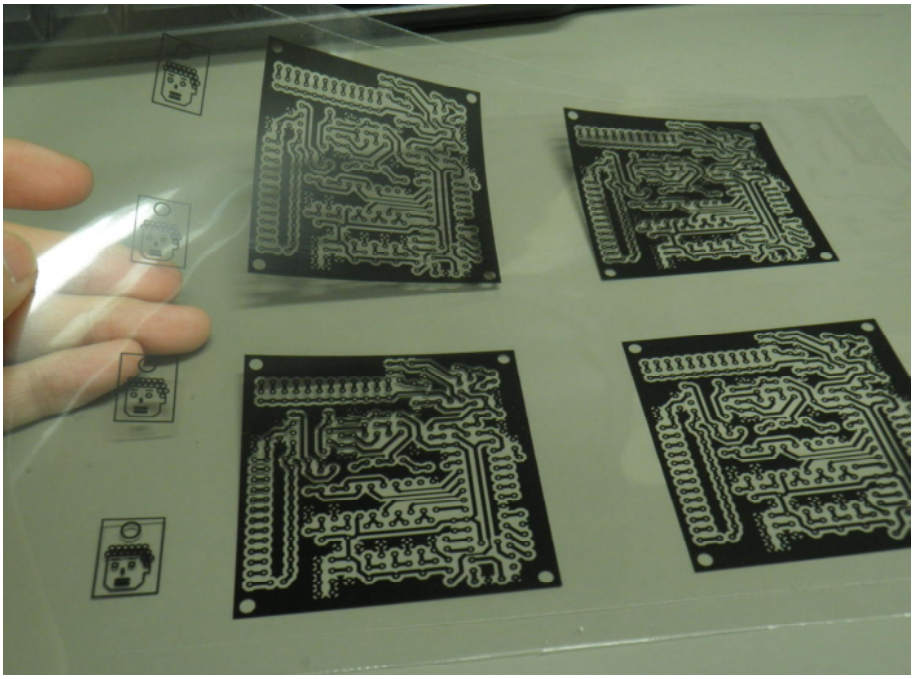

図 25　OHP フィルムに印刷したパターン図

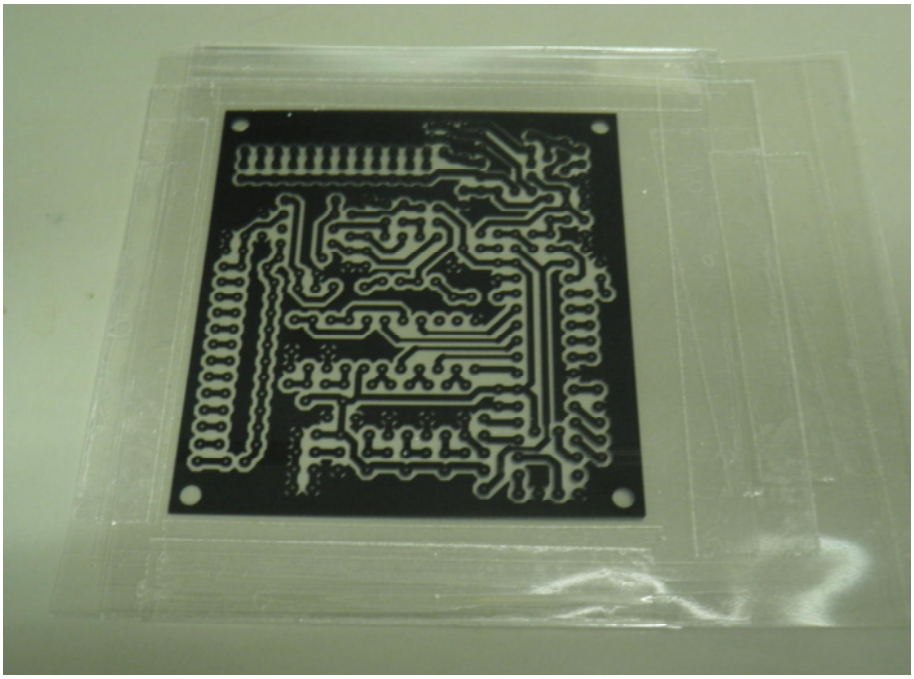

図 26　4 枚を貼り合わせた状態

②エッチング液

露光・現像を行ったあと，不要な銅を溶かす際に使用されるエッチング液・エッチング槽にも専用の製品が用意されている．専用のエッチング液は，反応が速く進むように工夫されているはずであるが，結局のところ主成分は塩化第二鉄である．本研究室では塩化第二鉄溶液[*2] を代替品として使用している．また，エッチング槽は沈殿物の処理が面倒であるため使用せず，100 円ショップで購入できるようなタッパを使っている．

図 27　塩化第二鉄溶液

*2 取扱，処分法には十分注意すること