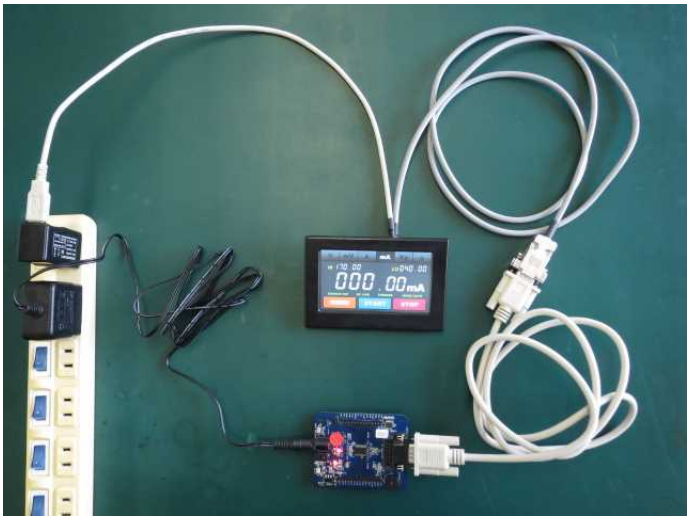
Smart-LCDC の使い方　－その１－

秋月電子の RL78 マイコンボードで 4.3 インチを動かす。



本書では、Smart-LCDC 完成品ユニット「UNT430C-01」を市販の RL78 マイコンボードと接続し、写真のような画面の動きを作るまでの手順を詳しく説明します。

このドキュメントで画面を作る基本をマスターできます。詳しい動画も準備していますので、弊社ホームページの「技術情報＞使い方のアドバイス集」をご参照ください。

2016/04/26

Ver1.00

はじめに

このドキュメントで Smart-LCDC で画面を設計する基本をマスターして頂くことができます。お絵かきツールが無いのが弊社の特徴ですが、逆にお絵かきツールが必要ないと言うほうが当たっていると思います。地道な作業に見えますが意外と簡単であることを体験してください。難しいことは何もありません。かえって、C 言語の自分の持っている知識だけで作画が行えるので、ツール等の余分な操作方法をおぼえる必要は全くありません。

本書の章１．２．は準備ですので、必ず目を通して頂くことをお勧めします。章３．４．は既にご存じの方は読み飛ばして頂いて構いません。章５．はチュートリアルと思って実施してみてください。章６．はデザインのやり方と絵の部品としての切り出しについて、章７．は実際のコーディングのやり方をとても丁寧に具体的に説明しています。

免責事項

本書や関連するビデオは参考書です。製品の動作を保証したりするものではありません。また本書を適用された製品の運用結果が如何なるものであっても弊社とは無関係です。実際の製品に応用される場合は自己責任で充分にご検証の上でお願い致します。

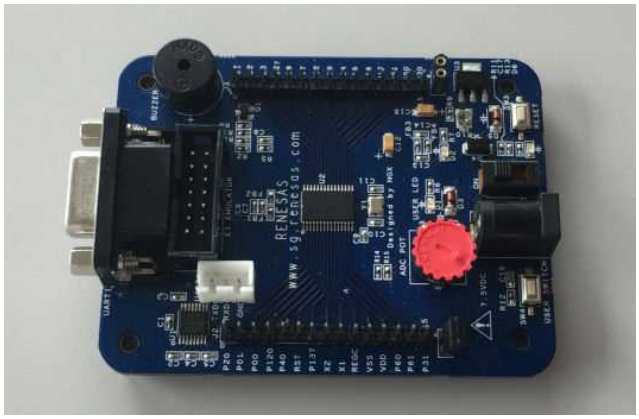
目次

１．必要なハードウェアの準備

本章では、デモに必要なハードウェアを紹介します。
デモに必要なものは、以下の通りです。

１）CPU ボード

・BlueBoard-RL78/G13_30pin-H（市販品、秋月電子通商にて 2400 円で購入可（2016/4/28 現在））
以下、「CPU ボード」と呼びます。



・上記用の電源。
9V 1.3A のスイッチング AC アダプター（秋月で適当なものを探しました）
型番　NP12-US0913

CPU ボードには 7.5V と記載がありますが、秋月電子の Web には 5V で動作確認済みと書いてあります。弊社では回路図を確認して 9V で動作させました。三端子の加熱もなく問題ないようです。

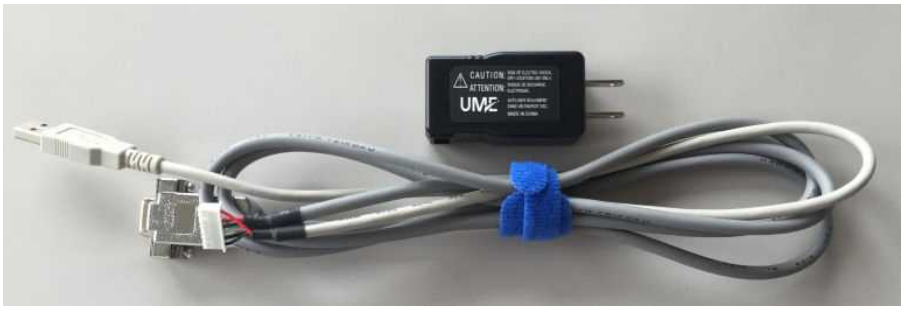
２）Smart-LCDC 完成品ユニット

・UNT430C-01（弊社製品の完成品ユニットです。STK430C-01 に含まれます。）


液晶側


裏面側

・KS-ELKIT（弊社製品の専用電源ユニットです。）



弊社のキット専用電源キットです。

3）ソフトウェアデバッガ

・Renesas E1（市販品、秋月電子通商にて 12600 円で購入可（2016/4/28 現在））

4）その他周辺機器

・microSD カード又は microSDHC カード（市販品）

SONY の SR-4A4（microSDHC カード、4GB、CLASS4）で動作確認
しています。　他何でも OK ですが、microSD カードでは画像の読み
出しスピードが遅いです。
microSD リーダー/ライターなどで、PC との接続手段も用意してください。

・D-Sub 9Pin インタリンク・クロスケーブル（市販品）

オス/オスの、pin2-3、7-8 がクロスしているものを使ってください。
カモンの 4AUS-GALE で動作確認しています。

・USB-シリアルコンバータ（市販品。ELECOM の UC-SGT で動作
確認しています。）

２．CPU ボードのジャンパ配線

本章では、CPU ボードの準備について説明します。

秋月電子の CPU ボード“BlueBoard-RL78/G13”を準備します。これには、RS-232C 用の D-sub　9 ピンコネクタが付いていますが、TXD1、RXD1、GND だけしかピン割り振りが成されていません。

この 9 ピンに弊社の UNT430C-01 の RS-232C を接続したいのですが、弊社側に存在する「RTS」がこの CPU ボードには存在しません。

それを解消するには CPU ボードの UART0 に割り振っている RxD0 を I/O ポートとして使います。

この RxD0 を本来 9pin のコネクタに備わっているはずの 7 番ピンに直接配線してしまいましょう。下の図 2-1 を参照ください。ここは NC（未接続）になっていますので配線して問題ありません。

以下の回路図のようにジャンパ線１本を追加してください。パターンカットは必要ありません。

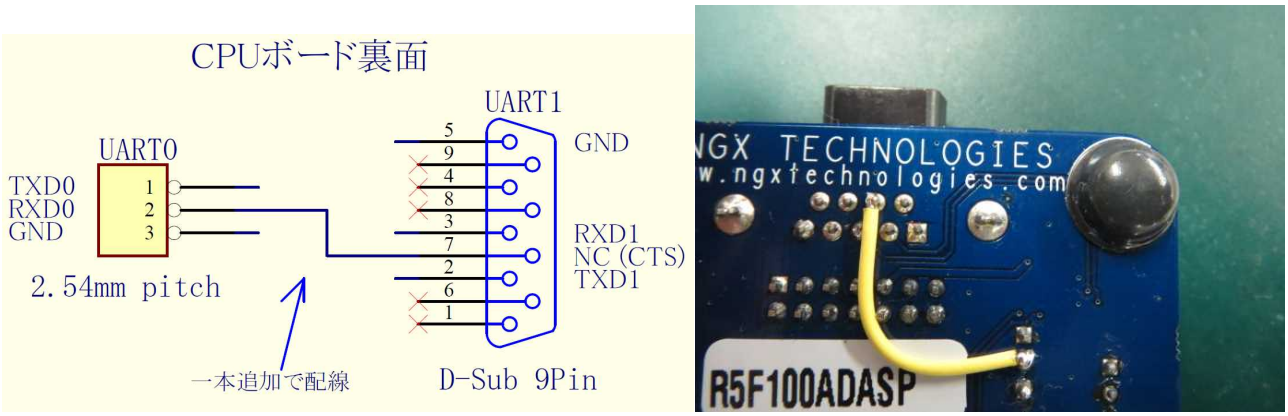


図２－１：ジャンパ配線部分

そして、CPU ボードと UNT430C-01 の接続を以下の図のように行ってください。

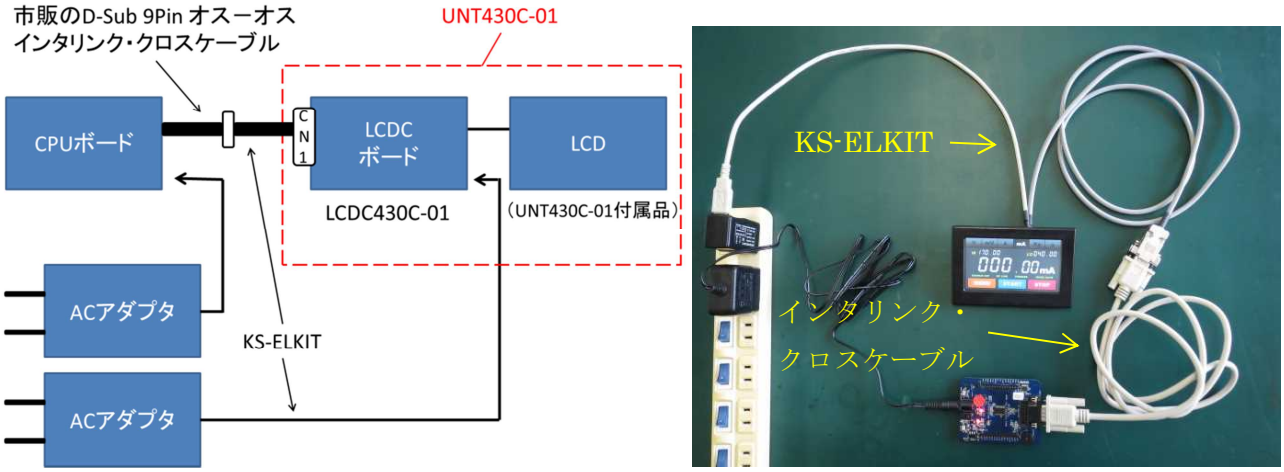


図２－２：CPU ボードと UNT430C-01 の接続図

３．ソフトウェアの準備

本章では、ソフトウェアの準備の方法を説明します。
ソフトウェアは、
・Renesas　　CS+ for CA,CX（開発環境）
・Renesas　　Renesas Flash Programmer（書き込みソフト）
・Tera Term（ターミナルソフト）
・計測画面サンプル一式（弊社 Web サイトにて提供）
が必要です。

１）開発環境のダウンロードとインストール

Renesas　　CS+ for CA,CX

以下の URL からダウンロード後、同社から提供されている手順に従ってインストールしてください。
（MyRenesas へのログインが必要です。）
http://japan.renesas.com/products/tools/ide/csp/downloads.jsp

図３－１　CS+ for CA.CX ダウンロード先
赤丸で囲んだ部分のような、CS+ for 「CA,CX」無償評価版の最新版をダウンロードしてください。

執筆時点で、V3.02.00 でした。以降はこれを用いて説明します。

Renesas　　Renesas Flash Programmer

以下の URL からダウンロード後、同社から提供されている手順に従ってインストールしてください。
（MyRenesas へのログインが必要です。）
http://japan.renesas.com/products/tools/flash_programming/rfp/downloads.jsp

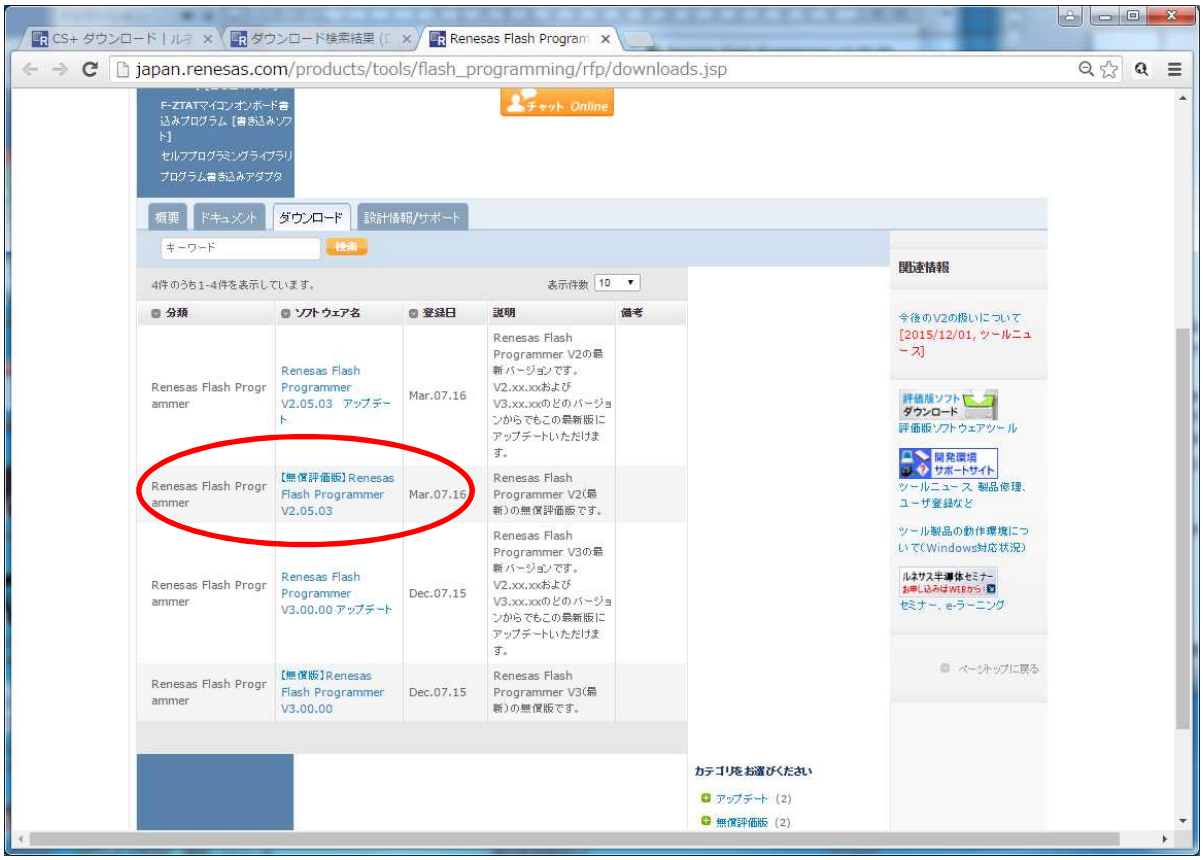


図３－２　Renesas Flash Programmer ダウンロード先

今回は、V2.xx.xx の無償評価版をダウンロードしてください。

執筆時点で、V2.05.03 でした。以降はこれを用いて説明します。

２）書き込みツールのダウンロードとインストール

Renesas Flash Programmer

メーカーから提供されている初期設定を終えた Renesas E1 を PC に接続してから、Renesas Flash Programmer を起動してください。

「ようこそ！」という画面が表示されるので、「新しいワークスペースの作成」「Basic モード」を選択して「次へ」をクリックしてください。

図３－３：「ようこそ！」画面

「使用するターゲット・マイクロコントローラ」に R5F100AD を選択し、「ワークスペース名」及び「プロジェクト名」に任意の名前、「作成場所」に任意の場所を入力して「次へ」をクリックしてください。図３－４のように、フィルタに「R5F100AD」を入力すると、スムーズです。

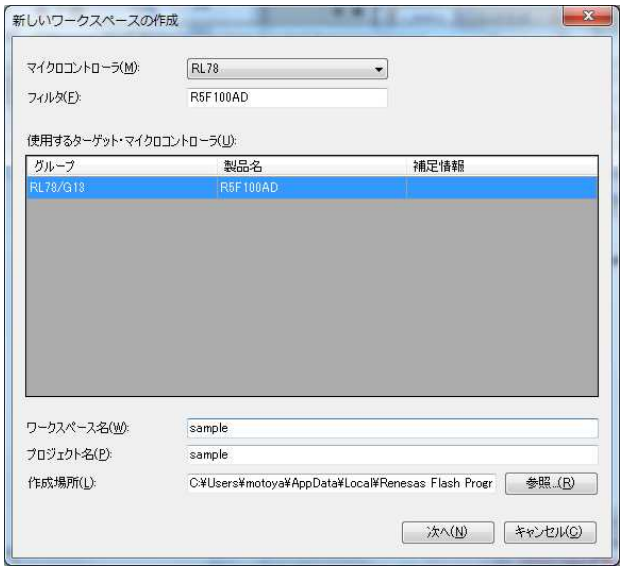


図３－４：「新しいワークスペースの作成」画面

使用ツールに「Renesas E1」が選択されていることを確認して「次へ」をクリックしてください。

図３－５：「通信方式」画面

このまま「次へ」をクリックしてください。

図３－６：「クロック供給」画面

「エミュレータから電源供給をする」にチェックを入れ、「供給電源」に「5.0V」を選択してください。

図３－７：「電源」画面

「完了」をクリックしてください。

図３－８：「プロジェクト設定機能一覧」画面

以上で、Renesas Flash Programmer の設定は完了しました。

図３－９の画面が表示されます。

「ユーザ/データエリア」横の「参照」をクリックして書き込みたいプログラムを選択し、スタートをクリックすることで、プログラムの書き込みを開始できます。

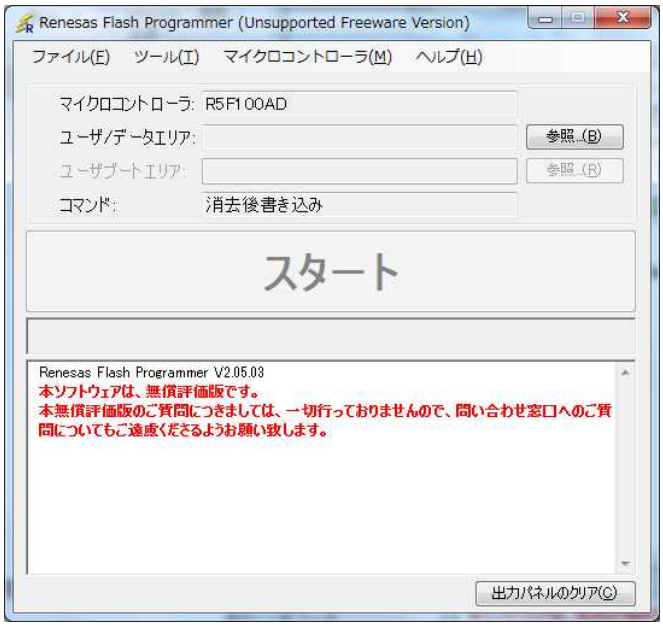


図３－９：ルネサスフラッシュプログラマの Basic mode

次回以降は、「ようこそ！」画面で「作成済みのワークスペース」を選択し、ここで制作したプロジェクトを選択することで設定を読み出せます。

３）ターミナルソフトのインストール

Tera Term

窓の社などでダウンロード後、インストールを行ってください。
検索エンジンで Tera Term を検索すれば､ダウンロード可能な Web サイトがいくつか見つかるはずです。
普段使い慣れているターミナルソフトがベストですが、ここでは Tera Term を使用して説明します。

Tera Term を使用する場合の設定を説明します。
PC と市販の USB-シリアルコンバータを接続するなどして、シリアルポートを確保してください。
Tera Term を起動すると、図３－１０のウィンドウが表示されます。
「シリアル」にチェックを入れ、確保したシリアルポートを選択してください。

図３－１０：シリアルポートの選択

設定　→　シリアルポートを選択して、シリアル通信の設定を行います。
図３－１１のように設定してください。
・ボーレート　：　115200bps
・フロー制御　：　hardware

図３－１１：Tera Term のシリアル通信設定

設定　→　端末を選択し、端末の設定を行います。
図３－１２の画面が表示されますので、以下の様に設定してください。

・改行コード　受信　：　CRLF
・「ローカルエコー」にチェック
・漢字―送信　：　SJIS

図３－１２：端末の設定

以上で、設定は完了しました。
設定　→　設定の保存を選択すると図３－１３のウィンドウが現れるので、ファイル名を付けて設定を保存してください。

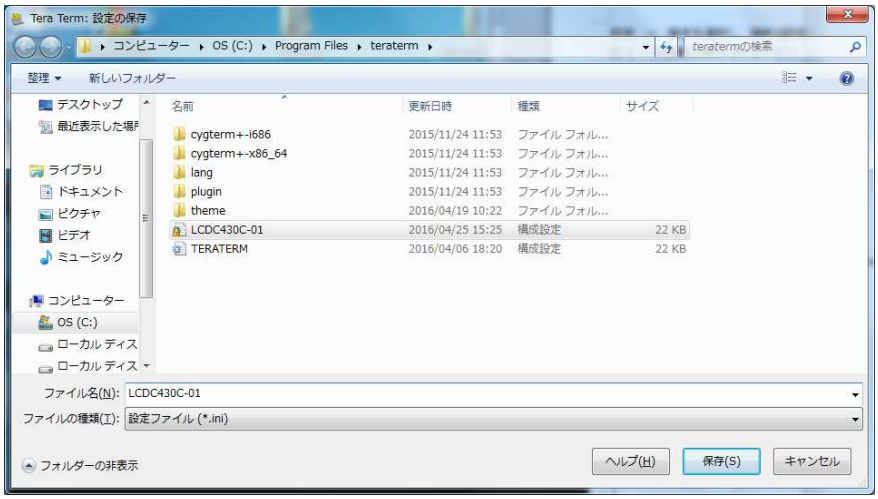


図３－１３：設定の保存

次回以降は、設定　→　設定の読み込みを選択すると設定を読み出すことができます。

４）計測画面サンプルのダウンロード、展開（解凍）

弊社 Web サイトから、計測画面サンプル一式をダウンロードしてください。
サンプル一式を解凍後、下記のディレクトリがあることを確認してください。

・pixsbmp
・RL78G13_V4Digitalmeters_RTS_SD
・RL78G13_V4Digitalmeters_RTS_FLASH
・RL78G13_V4Digitalmeters_noRTS_FLASH
・RL78G13_V4Digitalmeters_step1_FLASH
・RL78G13_V4Digitalmeters_step2_FLASH
・RL78G13_V4Digitalmeters_step3_FLASH
・RL78G13_V4Digitalmeters_step4_FLASH
・RL78G13_V4Digitalmeters_step5_FLASH

４．UNT430C-01 に画像を登録

本章では、デモに表示される画像データを、UNT430C-01 が扱えるようにします。
市販の画像編集ソフトで作成した標準の 24bit 色画像を bmp で保管しておけば、その画像をそのまま表示させることができます。手順は以下の流れです。

１．画像を表示させたい大きさで編集して保存します。
２．microSD カードに転送します。上限 8192 枚の絵のファイルまで処理できます。
３．「W1」コマンドで指示し、microSD カード内でそのまま表示できる画像形式に自動変換します。
４．「W0」コマンドで指示し、microSD カードからさらに LCD コントローラ管理下のシリアルフラッシュメモリに画像を転送します。

これらを順番に詳しく説明します。まず、
PC と UNT430C-01 を、市販の USB-シリアルコンバータを用いて下図のように接続してください。

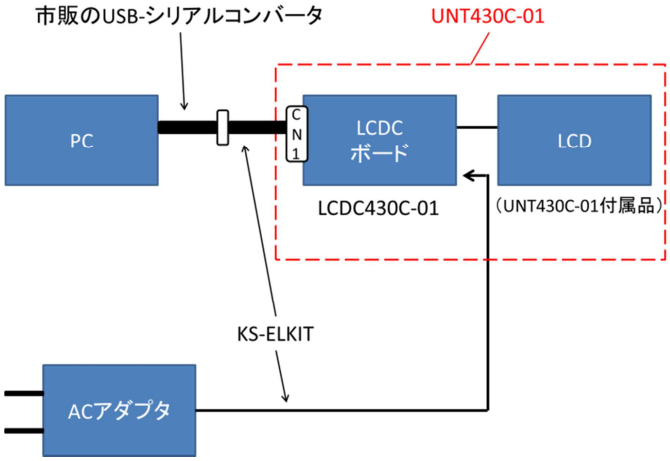


図４－１： PC と UNT430C-01 の接続図

ピン番号	仕様
1	+5V 電源
2	オープン
3	TXD
4	RTS
5	RXD
6	GND
7	GND

表４－１：CN1 の仕様

シリアル通信の設定が表４－２のようになっていれば、通信は可能です。ターミナルソフトの設定をこのように行ってください。

設定項目	仕様
ボー・レート	115200bps
データ長	8bit
パリティ	なし
ストップビット長	1bit
フロー制御	Hardware

表４－２：RS-232C シリアル通信の仕様

１）　画像の準備

画像を市販のソフトで編集したり写真を準備したりしますが、ここでは弊社のサイトからダウンロードしたデモ用の画像を使います。microSD カードから直接表示する場合は上限 8192 枚まで表示できます。LCD コントローラの管理下のシリアルフラッシュメモリに転送する場合は、容量が許す上限によって書き込める画像データ数が少なくなることがあります。

２）　準備した画像を microSD カードへ転送する

microSD カードを PC に挿入してください。
挿入した microSD カード内に”pixsbmp”という名前のフォルダを作成してください。
作成した”pixsbmp”フォルダ内に、デモ画像データをコピーしてください。
デモ画像データは、章３．の「４）計測画面サンプルのダウンロード、展開（解凍)」でダウンロード、展開した計測画面サンプルの「pixsbmp」ディレクトリに有ります。

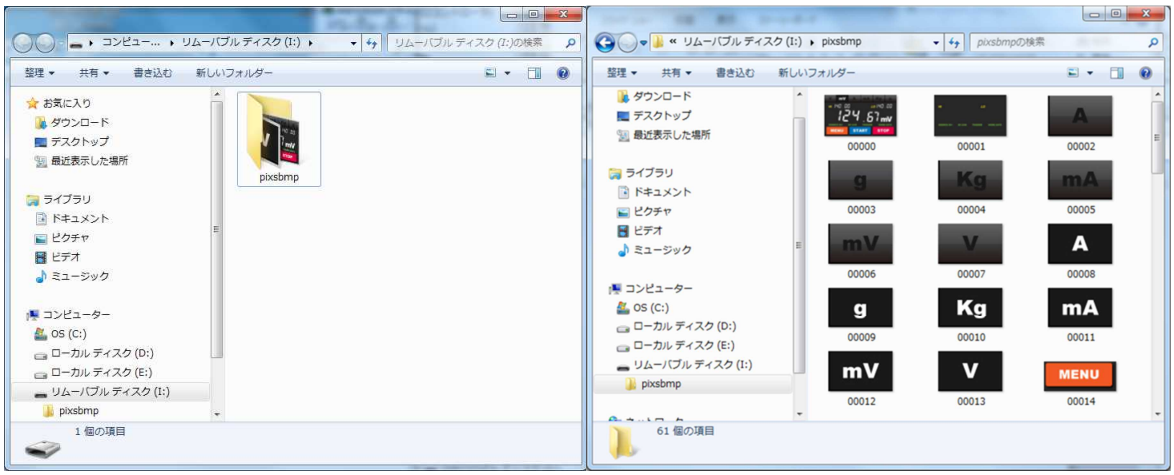


図４－３：microSD カードに”pixsbmp”フォルダを作り、画像をコピーする

３）　画像の変換と microSD カード内への保管

PC から microSD カードを取り外し、UNT430C-01 の電源が OFF 状態であることを確認し、microSD カードを microSD カードコネクタ（CN2）に差し込んでください。

UNT430C-01 の電源を投入後、Tera Term から「W1」と入力後にエンターキーを押下すると、UNT430C-01 の画面に「フォルダーを作成中です。」と表示されます。
SD カードに画像が登録できると、「SD カードへの書き込みを終了します。」と表示されるので、電源を

一旦切ってください。

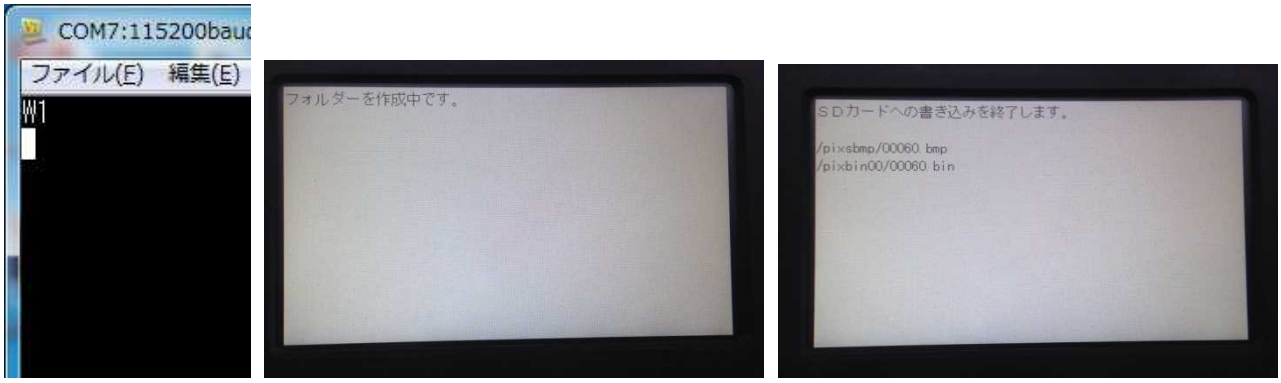


図４－４：SD カードに画像変換して登録する

UNT430C-01 の電源を再投入し、microSD カードに書き込んだデータを動作させてみます。
Tera Term から「P10000000000」と入力後にエンターキーを押すと、図４－５のような画面が現れます。

図４－５：デモソフト用画像「0000」

コマンドの意味は、P1 000 000 0000 の並びで
P1　microSD カードのデータを表示する
000　X 座標
000　Y 座標
0000　画像のシリアル番号

上記の座標位置を変えたり、画像のシリアル番号を変えたりして試してみてください。

４）　シリアルフラッシュメモリへの転送

UNT430C-01 の電源を投入後、Tera Term から「W0」と入力後にエンターキーを押下すると、UNT430C-01 の画面に「SFM を消去中です。」と表示されます。
シリアルフラッシュメモリに画像が登録できると、「SFM への書き込みを終了します。」と表示されるので、電源を切ってください。

図４－６：シリアルフラッシュメモリに登録する

UNT430C-01 の電源を再投入し、シリアルフラッシュメモリに書き込んだデータを動作させてみます。
UNT430C-01 に対し、「P00000000000」で画像表示をさせると、図４－５と同様の画面が表示されます。

コマンドの意味は、P0 000 000 0000 の並びで
P0　シリアルフラッシュメモリのデータを表示する
000　X 座標
000　Y 座標
0000　画像のシリアル番号

シリアルフラッシュメモリからの画像読み出しであるため、microSD カードを UNT430C-01 から取り外しても、画像を表示できます。
（注：microSD カードの抜き差しは、必ず UNT430C-01 の電源を切った状態で行ってください。）

これらの操作によって、UNT430C-01 は microSD カード、シリアルフラッシュメモリから画像を読み出して使用できるようになります。

余談ですが、これらの一連の作業は、すべてケニックシステム製のワンチップ LCD コントローラに予め備えている機能で実現しています。

５．表示デモの実行

本章では、先ほど UNT430C-01 が利用できるようにした画像を使い、２つの表示デモを実行します。

１）microSD カードから画像を読み出す場合

microSD カードから画像を読み出す場合のデモソフトは、ダウンロード、展開した計測画面サンプルの「RL78G13_V4Digitalmeters_RTS_SD」ディレクトリに配置されています。

ディレクトリ内の「RL78G13_V4Digitalmeters.mtpj」を CS+ for CA,CX で起動してください。
ビルド→ビルド・プロジェクトを選択することで、同ディレクトリの DefaultBuild 直下に「RL78G13_V4Digitalmeters.hex」ファイルが生成されます。

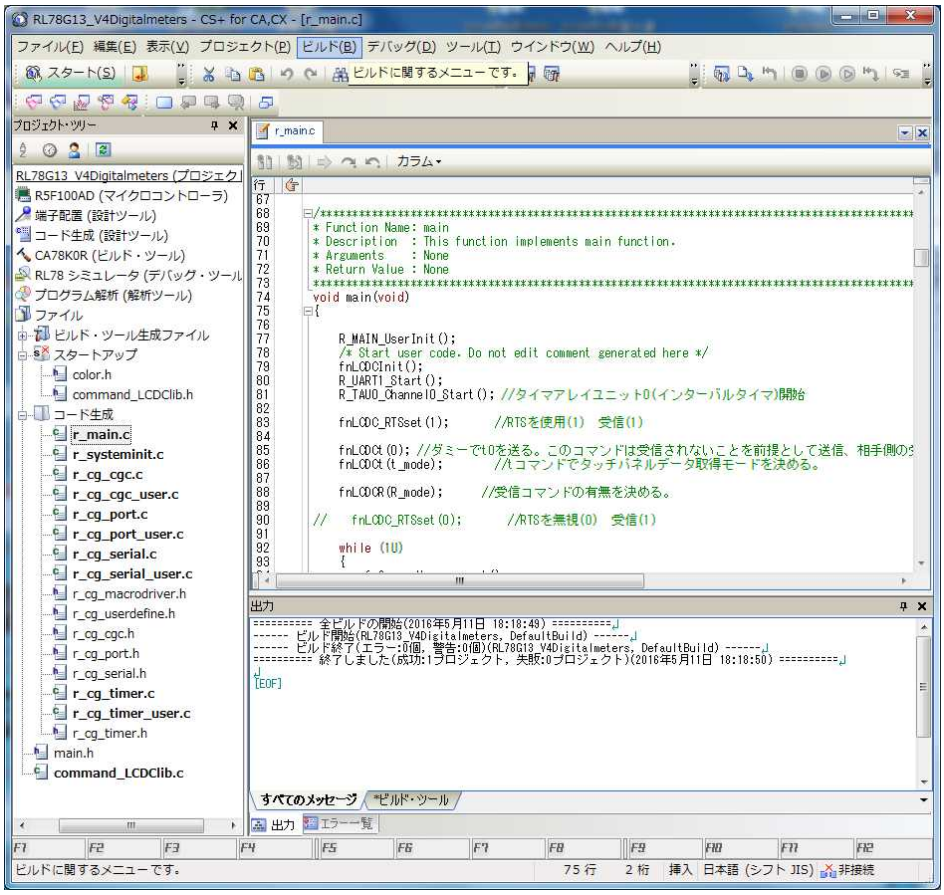


図５－１　ビルド終了

ビルドが終了したら、Renesas Flash Programmer にて、プログラムの書き込みを行います。
メーカーから提供されている初期設定を終えた Renesas E1 を PC に接続してください。
そして、Renesas Flash Programmer を起動して、マイコン「R5F100AD」に書き込みを行う設定を読み出してください。

Renesas Flash Programmer の設定を行っていなければ、章３「ソフトウェアの準備」を参考に設定してください。
マイコン「R5F100AD」を対象に書き込みを行う設定ができていれば OK です。

図５－２のメイン画面で、「ユーザ/データエリア」横の「参照」をクリックしてください。
ウィンドウが開くので、先ほど制作した「RL78G13_V4Digitalmeters.hex」を選択してください。
その後、Renesas E1 を CPU ボードに接続し、「スタート」をクリックすることで、プログラムの書き込みが行われます。

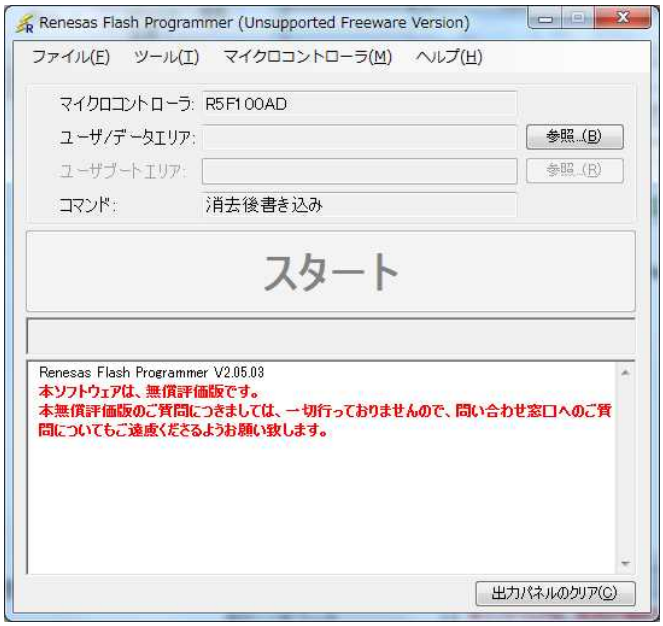
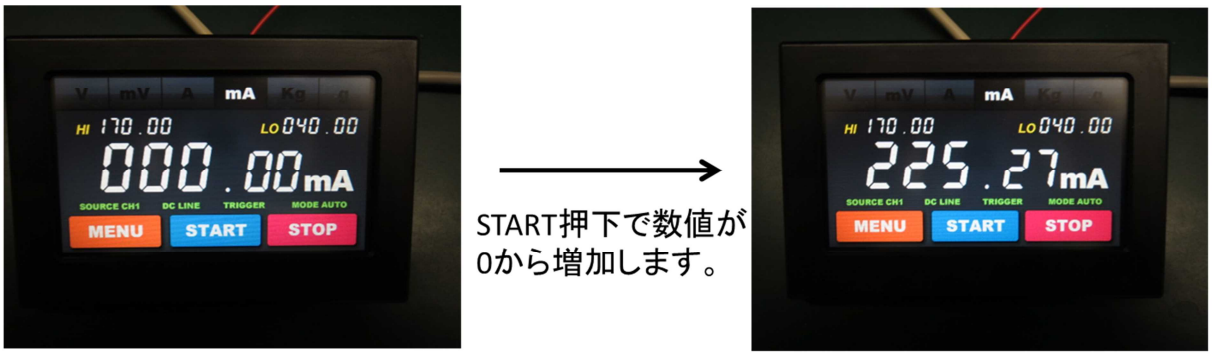


図５－２：ルネサスフラッシュプログラマの Basic mode

プログラムの書き込みが終了したら、Renesas E1 を CPU ボードから取り外してください。
そして、UNT430C-01 と CPU ボードを接続し、電源を入れることでデモ動作が始まります。
CPU ボードは、SW2 を ON することで電源が入ります。

デモは、下図のフローで進みます。



ソフト開始時の画面です。

数値が225.27になる、または
STOP押下で数値が止まります。

画面上側にあるV,mV,A,mA,Kg,gを押下すると、
画面右下に表示されている単位が変わります。

図５－３：デモのフロー

２）シリアルフラッシュメモリから画像を読み出す場合

同様に、ダウンロード、展開した計測画面サンプルの「RL78G13_V4Digitalmeters_RTS_FLASH」ディレクトリにある「RL78G13_V4Digitalmeters.mtpj」を起動してビルドし、生成された hex ファイルを CPU ボードに書き込みます。

プログラムの書き込みが終了したら、RenesasE1 を取り外してください。
UNT430C-01 と CPU ボードを接続し、電源を入れることでデモ動作が始まります。
シリアルフラッシュメモリから画像を読み出しているため、UNT430C-01 から microSD カードを取り外してもデモ動作を行うことができます。
（注：microSD カードの抜き差しは、必ず UNT430C-01 の電源を切った状態で行ってください。）

６．画面設計の実際（画像準備）

本章では、UNT430C-01 での開発をより深く知っていただくため、計測画面サンプルの作り方を紹介します。

１）表示する画面を検討する（ラフ設計）

図６－１　ラフ設計

仕様については、以下のことを決めています。

１．７セグメント LED のイメージの数字が任意に変化すること。
２．大きな数字の単位をタッチパネルで変更できること。
３．START、STOP で数字の変化、停止を行えること。
４．MENU を押すと、ソフト次第で別の画面に移ることを想定すること。

２）デザイン画の作成

弊社のデザイナに上記のコンセプトをベースに絵を描いてもらいました。それが次のページの絵です。

図６－２　詳細なイメージ図

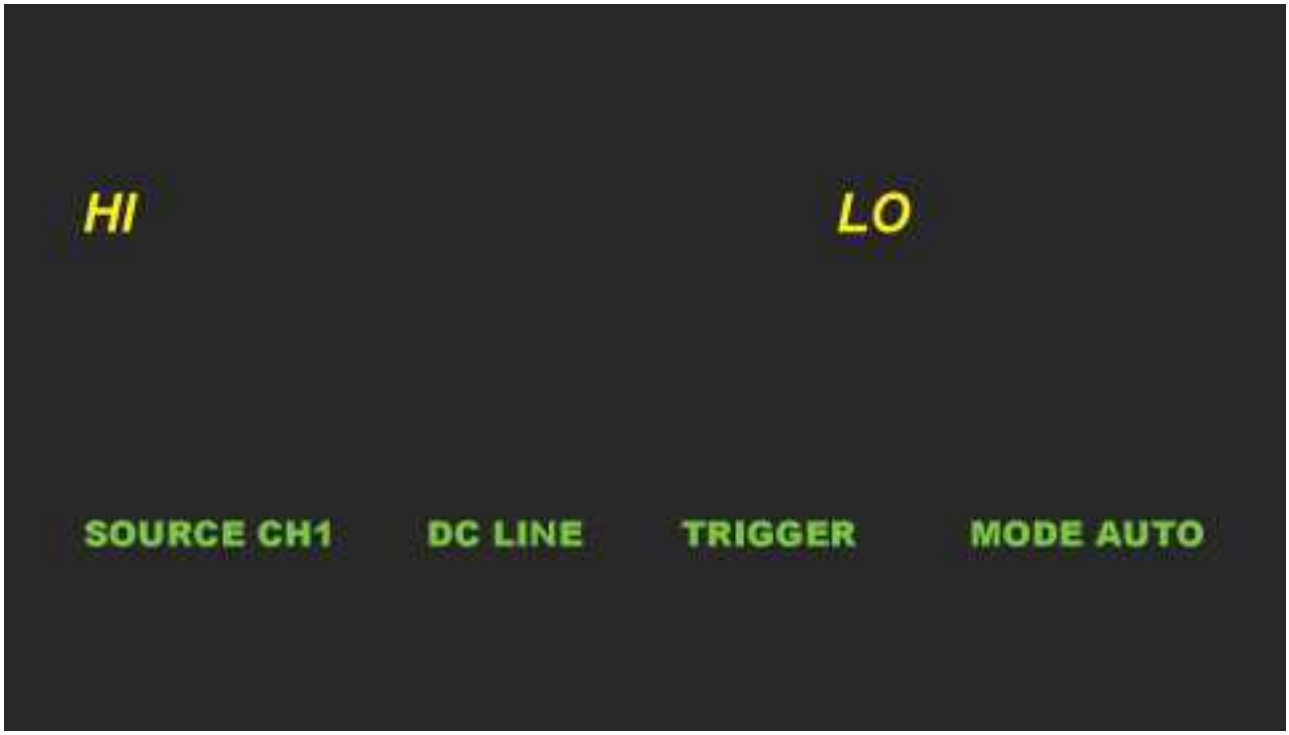
この設計図は動くところと固定のところをきっちり区別してデザイナに動く状態をイメージしてもらって絵を１枚描いてもらいます。
その完成図をプログラマーが確認し、基本的に長方形で囲まれた領域の書き換えで、画面の動きが本当に確保できるかどうかを確認します。簡単な確認だと思います。
それが OK なら次のステップに進みます。

次のステップはこの基本画面の部品化です。

たとえば、数字は 0～9 まで 10 種類必要です。もし極性や小数点が必要なら「＋」、「－」、「.」も作成します。
単位の V、mV、A 等も予め準備します。
タッチパネルの押しボタンにしたい場所は、格好良く見せるため、押したイメージと離したイメージを最低２つ準備します。さらにボタンに特別な表示機能や文字でさえ変化するオプションが必要なら、それらの画像も準備します。
弊社ではイラストレーターというソフトを使って作業をしていますが、予めデザイナーが作画したデザインだからと思いますが、複雑な絵であってもあとから大きな手直しをしたことはありません。そういう意味ではとても手離れが良いと思います。

３）絵の部品作成

以下のように順番に表示の部品作成をしてゆきます。



背景画

以下、個別部品を作画したものから切り出していって、下記のような部品を作成します。

注意すべきことは、切り出す部品画の位置と大きさを揃えて切り出していくということが必要です。

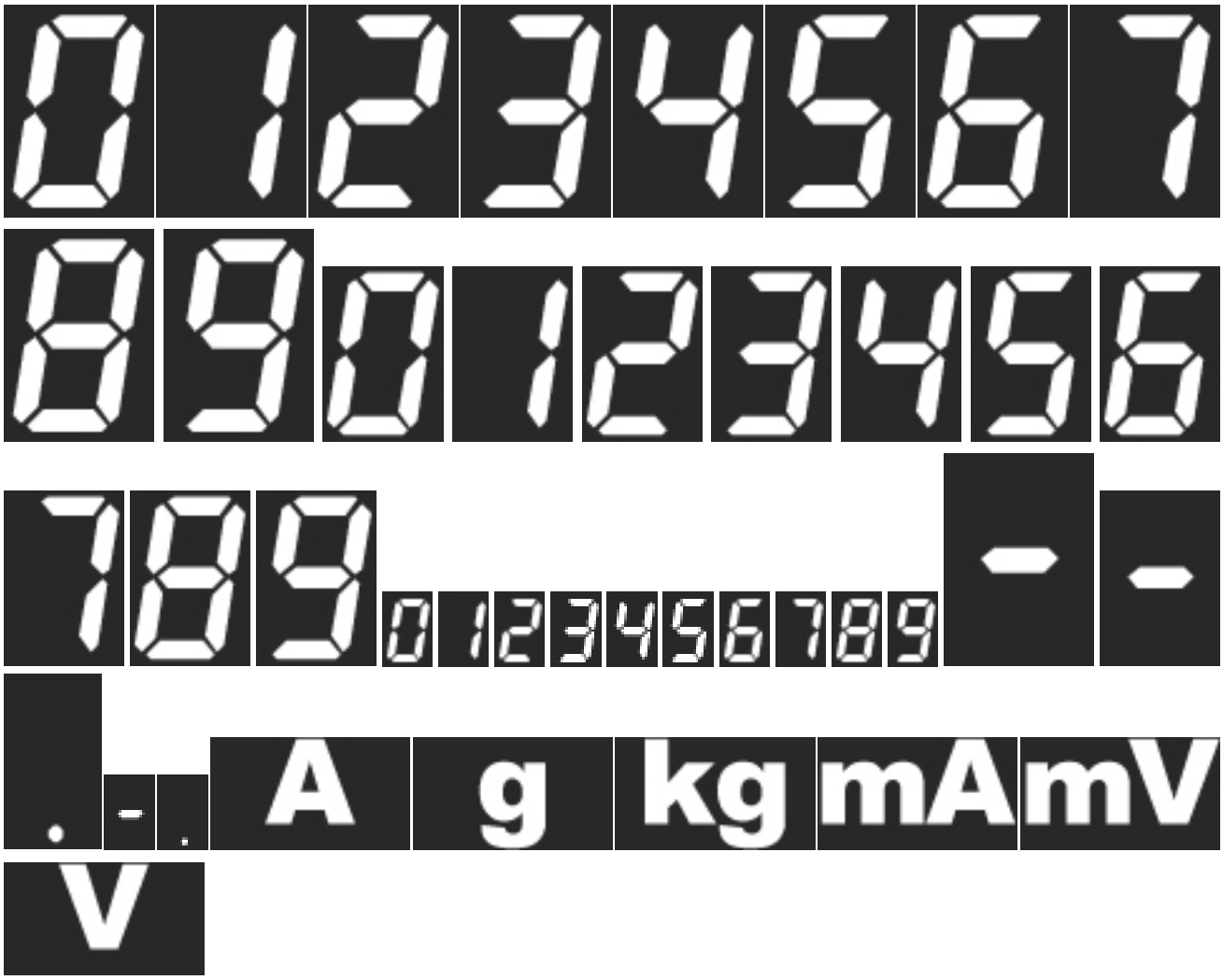


図６－３：ソフトに使用する画像（実物の大きさ）

以上、デザインコンセプトの最初の絵も含めて全部で 61 枚の画像ができました。

画像の名前は掲載している左上から順番に 00000.bmp から 00060.bmp と番号を振り分けています。

7 セグメントの数字画像に関して、番号を振り分けるとき、

・大数字：　7 セグ数字 0＝[00020.bmp]　～　7 セグ数字 9＝[00029.bmp]

・中数字：　7 セグ数字 0＝[00030.bmp]　～　7 セグ数字 9＝[00039.bmp]

・小数字：　7 セグ数字 0＝[00040.bmp]　～　7 セグ数字 9＝[00049.bmp]

などにすると、後々ソフトで制御しやすくなります。

4）各画像の表示位置を検討

左上を原点（X=0,Y=0）として、各画像の表示位置を決定します。画像を部品として切り出してくるときに、これらの位置情報はデザイナが把握していますので、予め一覧表にしてもらうといいと思います。

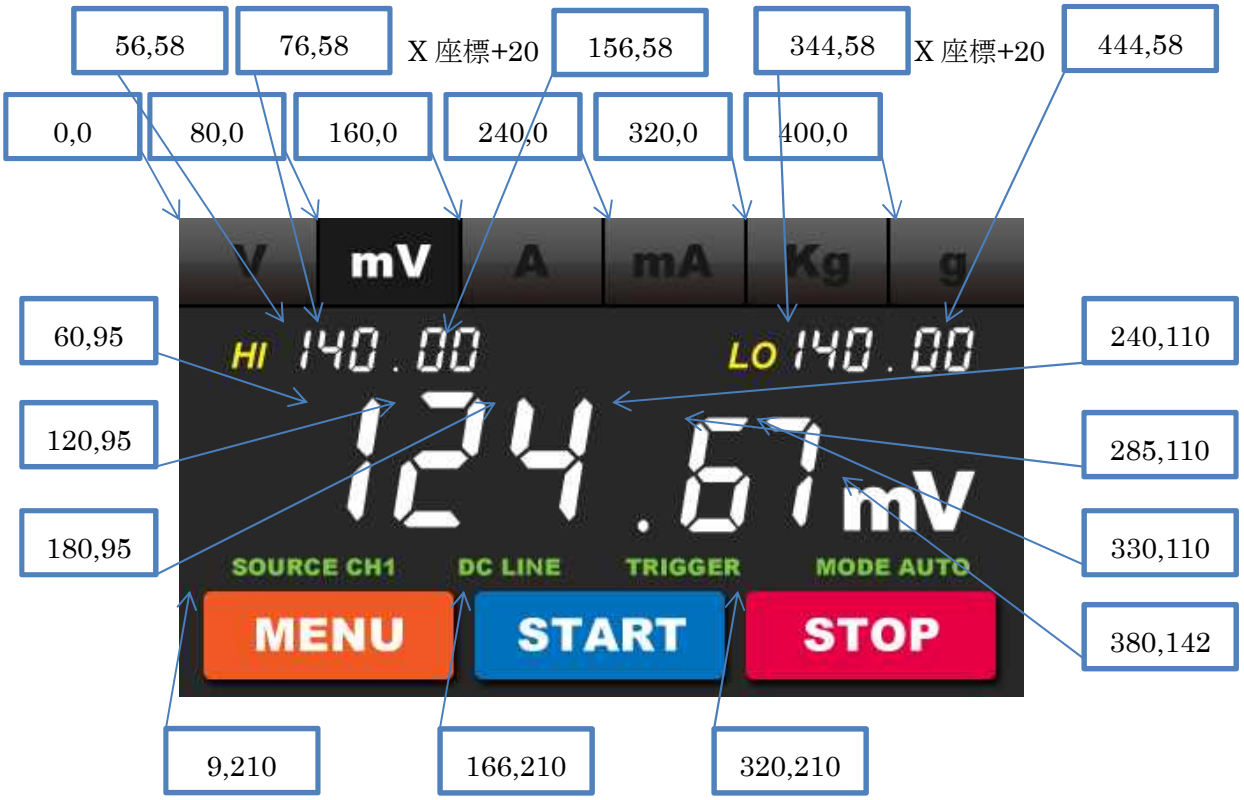


図６－４：各画像の表示位置

UNT430C-01 に登録したこれらの画像を、画面の任意の場所に呼び出して表示できるようになりました。

５）画像データの作成方法

この計測画面で表示される画像データは、この 4.3 インチの液晶のピクセルサイズ　480 x 272 pixel に合わせて作成しています。このサイズ以下の BMP ファイルを作ってください。3.5 インチ QVGA なら　320x240 pixel です。

Windows に標準インストールされているペイントツール、イラストレーター、ペイントショップ等の汎用画像編集ツールで、ファイルの種類を「24 ビットカラーのビットマップ」にして BMP ファイルとして保存したものを使用してください。

使用したい画像のファイル名は「00000.bmp」からスタートして「00001.bmp」「00002.bmp」…のように 5 桁の連続した 10 進数にしてください。
以上の法則に従った画像のファイル名の BMP ファイルを SD カードにコピーして、章４．で紹介した方法で UNT430C-01 が使用できるようにフォーマット変換します。

６）RTS 機能を使用する

UNT430C-01 の RS-232C 通信は、RTS 制御を行うことができます。
UNT430C-01 が受信可能である場合は CN1 の RTS ピンが L になりますので、
CPU ボードで RTS が L であることを読み取ったうえでコマンドの送信を行ってください。

RTS 制御を無視して CPU ボードから送信を行うソフトを用意しています。つまり、RTS の接続を省いたのと同等となります。
ダウンロード、展開した計測画面サンプルにある「RL78G13_V4Digitalmeters_noRTS_FLASH」ディレクトリ内のソフトです。
このソフトを実行すると、数値の上昇スピードは速いものの、UNIT430C-01 が多すぎるデータを受け取りきれず、画面の表示が間に合わなくなることが確認できるはずです。
この様子をご確認いただいて、このような表示でも問題のないアプリケーションであれば、RTS を省略できるでしょう。

７）タッチパネル機能を使用する

UNT430C-01 は、コマンドでタッチパネル機能を使用することができます。
章４．と同様の方法で、Tera Term から「t2」と入力してエンターキーを押してください。
その上で LCD の任意の場所に触れると、触れている間だけ触れた位置の座標の返送データが受信できます。

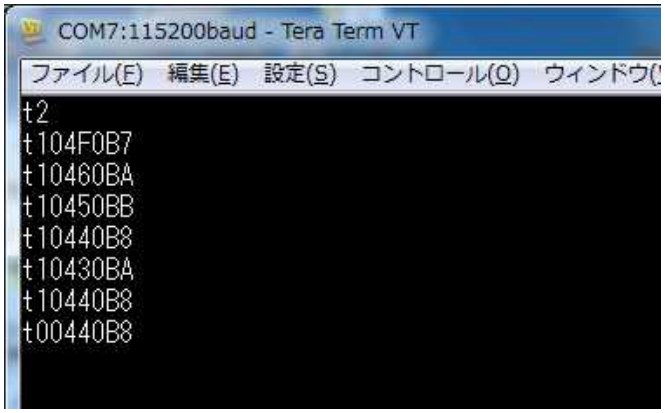


図６－５：「t2」実行後、LCD 左上を指で押した時の返送データ

タッチパネルデータ返送コマンドのコマンドフォーマットを示します。
「種別」によって返送データの送られる条件が異なり、今回の場合それは「2」です。
種別が 2 の場合、タッチパネルが押されている間のみ、タッチパネル座標の返送データが返送されてきます。

	コマンド	データ列
内容	t	種別
データ	1 バイト	1 バイト

表６－１：タッチパネルデータ返送コマンドのフォーマット

返送データのフォーマットを示します。

	コマンド	返送データ例		
内容	t	イベント	X 方向 AD 値	Y 方向 AD 値
データ	1 バイト	1 バイト	3 バイト	3 バイト

＊AD 値の分解能は 10 ビットです。
表６－２：返送データのフォーマット

フォーマット内の「イベント」は、'1'はタッチパネルが押されている状態、'0'はタッチパネルから指を離した状態を指します。
例えば、図６－５の「t104F0B7」返送データは、AD 値の座標（0x04F , 0x0B7）が押されていることを意味します。

詳細は、以下の弊社 Web サイトにある「KS350 430 570CT コマンドマニュアル」でご覧いただけます。
様々な方法でタッチパネル座標データを取得してみてください。
http://www.kenic.co.jp/download/index.html

７．画面設計の実際（ソフト設計による動きの作成）

前章で画像は準備でき、実現したい動きのイメージも頭にありますね。ではいよいよ準備した CPU ボードから UNT430C-01 を本格的に動かしてみましょう。

ダウンロード､展開した計測画面サンプル内に､下記の step1 ～ step5 を行ったソフトを用意しました。「RL78G13_V4Digitalmeters_step1_FLASH」～「RL78G13_V4Digitalmeters_step5_FLASH」が、step1 ～ step5 に対応しています。

今回のソフトは、CS+ for CA,CX　V3.02.00 で制作しています。

１）　step1　背景表示を確認

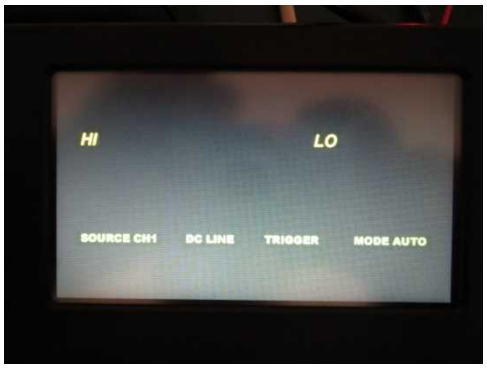


図７－１　「RL78G13_V4Digitalmeters_step1_FLASH」の画面

まずは、背景（00001.bmp）を座標[0,0]に表示するだけのソフトを制作します。
そのために、UNT430C-01 に画像を表示するコマンドを打つ関数を作ります。

ここで重要なことは、以下の 2 つです。
・ASCⅡコード準拠で作成したコマンドを送信し、コマンドの最後には'CR'(0x0d)を送信する
・RTS が'L'になって、UNT430C-01 が受信可能状態になるまで待ってからコマンドを送信する

①ASCⅡコード準拠で作成したコマンドを送信し、コマンドの最後には'CR'(0x0d)を送信する

サンプルソフトの「command_LCDClib.c」から、UNIT430C-01 に画像を表示させるためのコマンドを送信する関数「fnLCDCP」のソースを示します。

この関数は、画像の番号、呼び出し位置を指定して、画像を画面に表示させるものです。

```
/*************************************************************************
    関数名
        fnLCDCP
    引数
        type      0：単一＜シリアルフラッシュロムから＞
                  1：単一＜microSDから＞
        uiX,uiY  原点のX,Y座標(画像の左上座標)
        number  画像番号
    戻り値
        なし
    説明
        マイクロSDから、またはシリアルフラッシュロムから画像を呼び、任意の場所に表示する。

*************************************************************************/
void fnLCDCP(unsigned int type,unsigned int uiX,unsigned int uiY,unsigned int number){
    string1="P";

    toASC1(type);
    strcpy(string2,ascbuf);

    toASC3(uiX);
    strcat(string2,ascbuf);

    toASC3(uiY);
    strcat(string2,ascbuf);

    toASC4(number);
    strcat(string2,ascbuf);

    fnMake_sum_and_Trans();
}
```

コメントにある「uiX,uiY 原点の X,Y 座標」には画像を表示させる位置（画像左上）の X,Y 座標が、「number 画像番号」には表示させたい画像の名前が入ります。
共に 10 進数の数値です。
fnLCDCP 関数では、”P”、引数 uiX、引数 uiY、引数 number、(0x0d)を順番に ASCⅡコード化して、最後に fnMake_sum_and_Trans 関数で RS-232C 通信で送信しています。

関数中の toASC(1～n)関数は、10 進数 n 桁の引数を 1 桁ずつ ASCⅡコードに変換し、上の桁から順番に n バイトの配列「ascbuf」に格納します。
fnMake_sum_and_Trans 関数内にて、配列「string1」、「string2」の内容を連続して送信し、最後に’CR’(0x0d)を送信する仕組みになっています。

UNT430C-01 は画像を表示するためのコマンドを持っており、そのコマンドを使用して fnLCDCP 関数を作っています。
詳しくは、スタータキットをご購入されたお客様に開示される「STK430C-01 スタータキットマニュアル」をご参照ください。

②RTS が’L’になって、UNT430C-01 が受信可能状態になるまで待ってからコマンドを送信する

先ほど少しだけ登場した、「fnMake_sum_and_Trans」関数のソースの一部を示します。

```
/*************************************
関数名
      fnMake_sum_and_Trans
引数：なし
戻り値：なし
説明：コマンドを送信する。
       RTSが有効な場合、RTSを読み取りLowになるまで待機する。
       チェックサムが有効な場合、チェックサムを作りコマンドの後に接続する。
*************************************/
void fnMake_sum_and_Trans(void)

 （中略）
     if(rts_flag == 1) {
          if(RTS_PIN == 1){
               LED_OUT1 = 1;
          }
          while(RTS_PIN == 1);
          LED_OUT1 = 0;
```

```
    }
（中略）
}
```

この部分では、RTS の確認をしています。
RTS は UNT430C-01 の CN1 の pin4 に出力されており、UNT430C-01 がコマンド受信可能状態である場合、'L'になります。
RTS が'L'出力の間、RTS_PIN = 0 となるようにしています。
UNT430C-01 の RTS が'L'になるまではソフトを while ループして、送信を待機する仕組みです。
また、RTS 機能が働いて CPU ボードが送信できなくなっている間、CPU ボードの USER LED が消灯する（LED_OUT = 1）ようにしています。
まだ LED は消灯しませんが、step5 のソフトで「START」スイッチを押すと確認できると思います。

結果としてここで制作したソフトは、RTS が L になるまで待機したうえで、
CPU ボードから RS-232C シリアル通信で、「P00000000001(CR)」と一回送信しただけの内容です。
それ自体は、章４で PC と UNT430C-01 を接続して行ったこととほぼ同じです。

２）　step2　数字表示関数を作ってみる

図７－２：「RL78G13_V4Digitalmeters_step2_FLASH」の画面

数字の画像を呼び出す位置をあらかじめ決めておき、その場所に任意の画像（数字）を呼び出せるようにします。
それぞれの数字の画像、「mA」の画像呼び出しは、１）step1 で作成した fnLCDCP 関数を使用します。この関数は、画像の番号、呼び出し位置を指定して、画像を画面に表示させるものでした。
この fnLCDCP 関数を使って、画面中央の、小数点を挟んだ 5 桁の数字を表示させる関数「fnMeasurement_Count」を作ってみます。
（これらの関数は、「r_main.c」に入っています。）

ここで重要なことは、以下の１つです。
・数字の画像自体の準備

① 数字の画像自体の準備

関数「fnMeasurement_Count」のソースを示します。画面中央の、小数点を挟んだ 5 桁の数字を表示させる関数です。

```
void fnMeasurement_Count(int count,int count2){

    keta3 = count%10;
    keta4 = (count/10)%10;
    keta5 = (count/100)%10;

    fnLCDCP(0,L_NUN3_X,L_NUN_Y,20+keta5);
    fnLCDCP(0,L_NUN2_X,L_NUN_Y,20+keta4);
    fnLCDCP(0,L_NUN1_X,L_NUN_Y,20+keta3);
    fnLCDCP(0,M_NUN2_X,M_NUN_Y,30+count2);   //小数点以下1桁目
    fnLCDCP(0,M_NUN1_X,M_NUN_Y,30+count2);   //小数点以下2桁目
}
```

この関数にある 5 つの fnLCDCP 関数のうち、前半 3 つに注目してください。
「20 + keta(5 ~ 3)」番の bmp ファイルを読み出しています。

章６－３「絵の部品作成」で、今回用意した bmp ファイルの番号を、

・大数字：　7 セグ数字 0＝[00020.bmp]　～　7 セグ数字 9＝[00029.bmp]
・中数字：　7 セグ数字 0＝[00030.bmp]　～　7 セグ数字 9＝[00039.bmp]
・小数字：　7 セグ数字 0＝[00040.bmp]　～　7 セグ数字 9＝[00049.bmp]

と設定していました。

このとき「20 + keta(5 ~ 3)」番の bmp ファイルを表示することは、「keta(5 ~ 3)」に代入した 1 桁の数字そのままを大数字で表示することになります。
同じく、後半 2 つの fnLCDCP 関数も、「count2」に代入した 1 桁の数字そのままを中数字で表示することができる仕組みになっています。

このように、使用する bmp ファイルの番号をあらかじめ工夫しておくと、UNT430C-01 を使用したソフト開発をより簡単に進めることができます。

３）　step3　タッチパネルの動作を試してみる

図７－３：「RL78G13_V4Digitalmeters_step3_FLASH」の画面

押下している間だけ CPU に押下された座標データを送り、その位置にドットを打ちます。
タッチパネルデータの返送の条件を決める関数、任意の位置にドットを打つ関数、タッチパネルデータを取得して結果を処理する関数が必要です。
ドットを打つ関数「fnLCDCDot」については、fnLCDCP 関数と同様の作り方をしているので、ここでは省略します。
（上記ドットを打つ関数は、「command_LCDClib.c」に入っています。）

ここで重要なことは、以下の 3 つです。
・タッチパネルデータ返送コマンドで、返送の条件を設定できる
・座標の返送データの内容を処理して、タッチパネルが押されているか離されているかを判断する
・座標の返送データの内容を処理して、押された位置の座標を知る

① タッチパネルデータ返送コマンドで、返送の条件を設定できる

まずはタッチパネルデータの取得条件を決める関数「fnLCDCt」を用意します。
（サンプルソフトでは、「command_LCDClib.c」に作っています。）
fnLCDCt 関数を示します。

```
/*******************************************************
        関数名  fnLCDCt

        説明  タッチパネルデータの取得条件の設定

＜種別＞
0:1データ返送
1:自動で連続返送
2:タッチパネルが押されている間連続返送
3:タッチパネルが押されたときだけ返送
4:タッチパネルが離されたときだけ返送
5:タッチパネルが押された時と離された時　両方のデータを返送
*******************************************************/
void fnLCDCt(unsigned int type)
{
    string1 = "t";

    toASC1(type);
    strcpy(string2,ascbuf);

    fnMake_sum_and_Trans();
}
```

fnLCDCt コマンドを一度実行すると、それ以降、CPU ボードは UNT430C-01 から、引数「type」の値によって異なる条件でタッチパネル座標の返送データを取得できます。
今回は main 関数で引数を「2」にして実行しているので、タッチパネルが押されている間のみ、タッチパネル座標の返送データが送信されてきます。

② 座標の返送データの内容を処理して、タッチパネルが押されているか離されているかを判断する。

UNT430C-01 のタッチパネル機能を、タッチパネルが押されている間のみタッチパネル座標の返送データが帰ってくるという設定にして使用する設定ができたと思います。

この動作はより詳しく説明すると、「タッチパネルが押されたらデータの自動的な連続返送を開始し、タッチパネルから指を離すと、離された時点の X 方向 AD 値及び Y 方向 AD 値を返送して返送動作を中断する。」という動作です。

タッチパネルのデータ返送の条件の詳しい仕様は、以下の弊社 Web サイトにある「KS-430CT-I1 ハードウェアマニュアル」をご参照ください。
http://www.kenic.co.jp/download/index.html

タッチパネル座標の返送データのフォーマットを示します。

	コマンド	返送データ例		
内容	t	イベント	X 方向 AD 値	Y 方向 AD 値
データ	1 バイト	1 バイト	3 バイト	3 バイト

＊AD 値の分解能は 10 ビットです。
表 7－1 ：返送データのフォーマット

フォーマットにある「イベント」は、タッチパネルが押されている場合'1'(0x31)となり、タッチパネルが離されている場合は'0'(0x30)となります。
つまり、今回の条件では、タッチパネルが離された瞬間のみ、「イベント」が'0'となる返送データが帰ってくることになります。

以上を踏まえると、タッチパネルが押されているか離されているかを判断したければ、最新の受信データの「イベント」が'0'なのか'1'なのかで判断すれば良さそうです。

サンプルソフトの「command_LCDClib.c」から、タッチパネルが押されているか離されているかを判断する「fnt」関数を示します。

```
/*********************************************************************
    関数名
        fnt
    引数
        ptr：受信したtコマンドが収納されているアドレス
    戻り値
```

```
            なし
        説明
            tコマンドの2文字目のタッチイベントを判断する。

**********************************************************************/
void fnt(unsigned char *ptr)
{
        ptr++;
        if(*ptr == '1') {
            //タッチパネルが押されているとき
            fntget(ptr);
        } else if(*ptr =='0') {
            //タッチパネルが離されているとき
//          fntget(ptr);
            usLCDCuiX = 0;
            usLCDCuiY = 0;
        } else {
        }
}
```

最新の返送データの配列のアドレスをこの関数の引数に入れると、「イベント」の値から、タッチパネルが押されている場合は座標の返送データ処理を行う「fntget()」関数に進みます。
タッチパネルから指が離れている場合は、座標データ usLCDCuiX、usLCDCuiY にとりあえず 0 を代入して関数を終了します。

③ 座標の返送データの内容を処理して、押された位置の座標を知る。

再び、タッチパネル座標の返送データのフォーマットを示します。

	コマンド	返送データ例		
内容	t	イベント	X 方向 AD 値	Y 方向 AD 値
データ	1 バイト	1 バイト	3 バイト	3 バイト

＊AD 値の分解能は 10 ビットです。

表 7 － 2 ：返送データのフォーマット

フォーマットにある「X 方向 AD 値」「Y 方向 AD 値」は、AD 値の 3 桁の 16 進数が、ASCⅡコードに従って 3 バイトの配列で表現されています。

例えば、AD 値 10（10 進数）は、「'0'」「'0'」「'A'」（0x30、0x30、0x41）の文字列として表現されます。

サンプルソフトの「command_LCDClib.c」から、座標の返送データを 10 進数の座標に変換する、「fntget」関数を示します。

```
/*********************************************************************
    関数名
        fntget
    引数
        ptr：受信したtコマンドが収納されているアドレス
    戻り値
        なし
    説明
        t1コマンドによって送られてきた座標データを
        (usLCDCuiX,usLCDCuiY)へ代入する。

*********************************************************************/
void fntget(unsigned char *ptr)
{
    char lcdcdataX[4],lcdcdataY[4];
    ptr++;
    lcdcdataX[0] = *ptr;
    ptr++;
    lcdcdataX[1] = *ptr;
    ptr++;
    lcdcdataX[2] = *ptr;
```

```
    ptr++;
    lcdcdataX[3] = '¥0';

    lcdcdataY[0] = *ptr;
    ptr++;
    lcdcdataY[1] = *ptr;
    ptr++;
    lcdcdataY[2] = *ptr;
    ptr++;
    lcdcdataY[3] = '¥0';

    usLCDCuiX = (asctoi(lcdcdataX) / 2) -10;
    usLCDCuiY = (unsigned short)(asctoi(lcdcdataY) / 3.7);
}
```

この関数の引数には、タッチパネル座標の返送データの 2 バイト目以降が入ります。
そのため、lcdcdatax には、返送データの X 座標部分 3 バイトの配列が、lcdcdatay には、返送データの Y 座標部分 3 バイトの配列が格納されています。

asctoi 関数は、引数とした 3 バイトの数列を 10 進数の数字に変える関数です。
返送データは 10bit の AD 値であり、最大値は 1024 です。
AD 値最大のデータが返送されてきた場合、
X 座標は usLCDCuiX = (1024 / 2) - 10 = 502、
Y 座標は usLCDCuiY = (1024 / 3.7) = 277 となります。

UNT430C-01 付属の LCD の画面は 480 x 272 dot なので、これで事実上問題のない誤差でタッチパネルの座標を取得できます。

４）　step4　タッチパネルで動くスイッチを作ってみる

図７－４：「RL78G13_V4Digitalmeters_step4_FLASH」の画面

タッチパネルで動作するスイッチを描画し、特に機能のないただのスイッチを作ってみます。
具体的には、画面下の「MENU」「START」「STOP」はスイッチを押している間だけ画像が切り替わり、画面上の「V」「mV」「A」「mA」「Kg」「g」は押されたスイッチだけが明るく見えるようにします。

ここで重要なことは、以下の１つです。
・画像を差し換えることで、スイッチが押された状態を演出する

①画像を差し換えることで、スイッチが押された状態を演出する
今回はあらかじめ、スイッチが押されている状況の画像、スイッチから指が離されている状況の画像を用意しました。
スイッチが押されたり離されたりするたびにこれらの画像を上書きすれば、スイッチを押したり離したりしているように見えます。

「r.main.c」から、main ループ処理を担当している「fnGazou_Measurement」関数の一部を示します。

```
void fnGazou_Measurement(void){

        （中略）

    if(switchRetu_UPEDGE & SWITCH2_UP){        //ｽｲｯﾁ2立ち上がりエッジ検出
        fnLCDCP(0,START_X,START_Y,17);      //START表示　押されてる方
    }

    if(switchRetu_DOWNEDGE & SWITCH2_UP){     //ｽｲｯﾁ2立下りエッジ検出
        fnLCDCP(0,START_X,START_Y,16);      //START表示　離した方
```

```
        }

        if(switchRetu_UPEDGE & SWITCH4_UP){        //ｽｲｯﾁ4立ち上がりエッジ検出
            if(sw4_data == 1){
                fnLCDCP(0,UNITSW1_X,UNITSW_Y,13);     //V  をおしてある画像にする。

                fnLCDCP(0,UNITSW2_X,UNITSW_Y,6);      //mV
                fnLCDCP(0,UNITSW3_X,UNITSW_Y,2);      //A
                fnLCDCP(0,UNITSW4_X,UNITSW_Y,5);      //mA
                fnLCDCP(0,UNITSW5_X,UNITSW_Y,4);      //kg
                fnLCDCP(0,UNITSW6_X,UNITSW_Y,3);      //g
            }
            if(sw4_data == 2){
                fnLCDCP(0,UNITSW2_X,UNITSW_Y,12);       //mVをおしてある画像にする。

                fnLCDCP(0,UNITSW1_X,UNITSW_Y,7);
                fnLCDCP(0,UNITSW3_X,UNITSW_Y,2);
                fnLCDCP(0,UNITSW4_X,UNITSW_Y,5);
                fnLCDCP(0,UNITSW5_X,UNITSW_Y,4);
                fnLCDCP(0,UNITSW6_X,UNITSW_Y,3);
            }
        （中略）
}
```

スイッチ 2 は、「START」スイッチです。
タッチパネルで START スイッチの位置が押された時を立ち上がりエッジ、指が START スイッチから離された時を立下りエッジとしました。
立ち上がりエッジ時は画像「00017」を上書きしてスイッチが押されたように、立下りエッジ時は画像「00016」を上書きしてスイッチから指が離されたように見せています。

スイッチ 4 は、画面上部にある 6 つのスイッチをひとまとめにしています。
「V」が押された時は変数 sw4_data = 1、「mV」が押された時は変数 sw4_data = 2 となる仕組みをこの手前に作ってあります。
スイッチが押されるたびに、押されていないスイッチは画像「00004」～「00009」を上書きし、押されたスイッチは「00010」～「00015」を上書きするようになっています。

図７－５：左上から順番に、画像「00004」～「00017」（大きさ 50%）

５）　step5　計測画面として完成させてみる

図７－６：「RL78G13_V4Digitalmeters_step5_FLASH」の画面

ここまで来れば、あとはスイッチを押したときの画面の動き方を自由に作るのみです。
UNT430C-01 のこれまでに紹介した機能で、多彩な動作の計測画面が作れるはずです。

今回は、以下の例に従って制作しました。

例：青いスイッチ「START」
　　大きな表示の 7 セグを 000.00 にクリア後、225.27 までカウントアップさせる。
例：赤いスイッチ「STOP」
　　カウントアップ中に押すことで、カウントアップを停止する。
例：オレンジのスイッチ「MENU」
　　押すことで、画面を初期化する。（カウントアップ時は押されても無効）
例：上の「V」「mV」「A」「mA」「Kg」「g」
　　押すことで、7 セグの単位が変更する。（カウントアップ時は押されても無効）

スイッチや数字の表示位置を変えることもできますし、CPU ボード側に何らかの計測機器を搭載して計測結果を表示させるようにするのも自由です。
色々な動作をさせてみてください。