"てのりマイコン" STM32F4-Console　マニュアル

1:"てのりマイコン"について

てのりマイコン STM32F4-Console は、32 ビットマイコン、43 キーのキーボード、USB インタフェイス、マイクロ SD コネクタ、BASIC インタプリタ、シリアルモニタ、3 チャンネルの ADC 入力を搭載し、モノクロ 400x240 解像度ビットマップ液晶と、USB から充電できるバッテリーを搭載可能な、てのひら動作をするマイコンです。

BASIC インタプリタはエディタ内蔵で、どこでもプログラミングが可能です。BASIC からマイクロ SD 中のファイルを読み書きすることも、シリアル通信を制御することもできます。

本基板には更に Bluetooth モジュールと RS-422 シリアル通信インタフェイスが増設可能です。Bluetooth モジュールは他の Bluetooth 機器と SPP で通信します。

それらを実装されたい方は、別紙の組み立て説明書に従って、部品を入手して半田付けしてください。また RS-422 インタフェイスも増設可能となっています。

2:必要な部品

以下の部品をはんだ付けしてください。

	品名	種類	サイズ	購入先	価格
U1	STM32F405RGT6	MPUARM Cortex-M4 64Pin		Digi-key	1184 円
U3	MAX1555	LiPo バッテリ充電制御 IC		Digi-key	251 円
U4	TAR5S33	3.3V レギュレータ		秋月	10 個 280 円
D1	MBR0520LT	ショットキーバリアダイオード		秋月	20 個 300 円
D2,D3,D4,D5,D6	1SS352	ショットキーバリアダイオード		秋月	40 個 200 円
U2	DIM3AT-SF-PEJ	MicroSD カードソケット		秋月	140 円
X1	MUSB-5B-NE-S175	USB MiniB 基板実装コネクタ		秋月	2 個 100 円
L1,L2,L3,L4	BLM21PG331	チップインダクタ		秋月	25 個 100 円
Q1	8MHz	水晶発振子 HC49/S 型		秋月	10 個 400 円
Q2	32.768kHz	水晶発振子 TC26H 型		秋月	4 個 100 円
U6	FH12-10S-0.5SH	液晶フラットケーブルコネクタ		秋月(液晶に付属)	3000 円
LED1,2,3		チップ LED	1608	秋月	20 個 200 円
PushSW 1～43	SKRPACE010	タクトスイッチ		秋月	5 個 100 円
C1,C2	22pF	積層セラミックコンデンサ	1608	千石	10 個 50 円
C3,C4	7pF	積層セラミックコンデンサ	1608	鈴商	10 個 100 円
C9,C16,C20,C23,C25,C26,C27,C28	0.01uF	積層セラミックコンデンサ	1608	千石	10 個 50 円
C5,C10,C11,C17,C29	0.1uF	積層セラミックコンデンサ	1608	千石	10 個 50 円
C39	0.33uF	積層セラミックコンデンサ	2012	鈴商	10 個 100 円
C8	1uF	積層セラミックコンデンサ	1608	秋月	100 個 200 円
C13,C18,C19	1uF	積層セラミックコンデンサ	2012	鈴商	10 個 100 円
C6,C7,C15	2.2uF	積層セラミックコンデンサ	2012	鈴商	10 個 100 円
C14	4.7uF	積層セラミックコンデンサ	2012	秋月	40 個 200 円
C30	10uF	積層セラミックコンデンサ	3216	秋月	8 個 100 円
C12	4.7uF	積層セラミックコンデンサ	3225	秋月	5 個 200 円
R5,R6,R7,R8,R9,R10	22Ω	抵抗	1608	千石	10 個 50 円
R24,R25	33Ω	抵抗	1608	千石	10 個 50 円
R2,R3	27Ω	抵抗	1608	千石	10 個 50 円
R29,R38,R39,R42,R43,R45	100Ω	抵抗	1608	千石	10 個 50 円
R16,R18	150Ω	抵抗	1608	千石	10 個 50 円
R1,R22	220Ω	抵抗	1608	千石	10 個 50 円
R23,R31	470Ω	抵抗	1608	千石	10 個 50 円
R21	510Ω	抵抗	1608	千石	10 個 50 円
R27,R28	1kΩ	抵抗	1608	千石	10 個 50 円
R4	1.5kΩ	抵抗	1608	千石	10 個 50 円
R38,R39	2.7kΩ	抵抗	2012	鈴商	10 個 100 円
R19	5.1kΩ	抵抗	1608	千石	10 個 50 円
R20,R26,R30	10kΩ	抵抗	1608	千石	10 個 50 円
R11,R12,R13,R14,R15,R17	51kΩ	抵抗	1608	千石	10 個 50 円

その他、動作には液晶とLiPoバッテリが必須です。USBをつないだ状態でも動作しますが、それではあまり面白くないでしょう。本マイコンはカレンダークロックを持っていますが、これを使うためには1220か1225のボタン電池と電池ホルダが必要です。

秋月電子で売っている、シャープ製液晶LS027B4DH01を、基板背面のフラットケーブルコネクタに接続します。このとき液晶は表面を向けた状態で接続しましょう。リボンケーブルを曲げて液晶を基板前面に持ってきて、ここで液晶を固定します。そのために液晶に両面テープを貼っておくといいでしょう。基板前面にはMPUやOPアンプなど部品があり段差になりますから、これら段差をカバーするような厚みのある両面テープを重ねて使うことになるでしょう。

フラットケーブルがきちんと接続していれば、USBケーブルを刺せば液晶に初期メッセージが表示され、BASICインタプリタの入力待ちになる筈です。

1セルのリチウムイオンかリチウムポリマ二次バッテリならどれでも使えます。定格3.7V以上なら他のバッテリでも動きますが充電はリチウム二次電池しか対応しません。

基板表面右上/裏面左上のBATT端子にバッテリを接続します。極性を間違えないでください。POWERスイッチを左に動かし、USB Mini-Bコネクタにケーブルを接続して充電します。充電中は基板裏の赤LEDが点灯しています。消灯したら充電終了です。

秋月電子で売っているものや、aitendoで売っている同じような電池ホルダをJP15/JP16にはんだ付けします。ボタン電池を入れると、この電池が電源を切っても内蔵カレンダーの内容を保持します。

ケース:

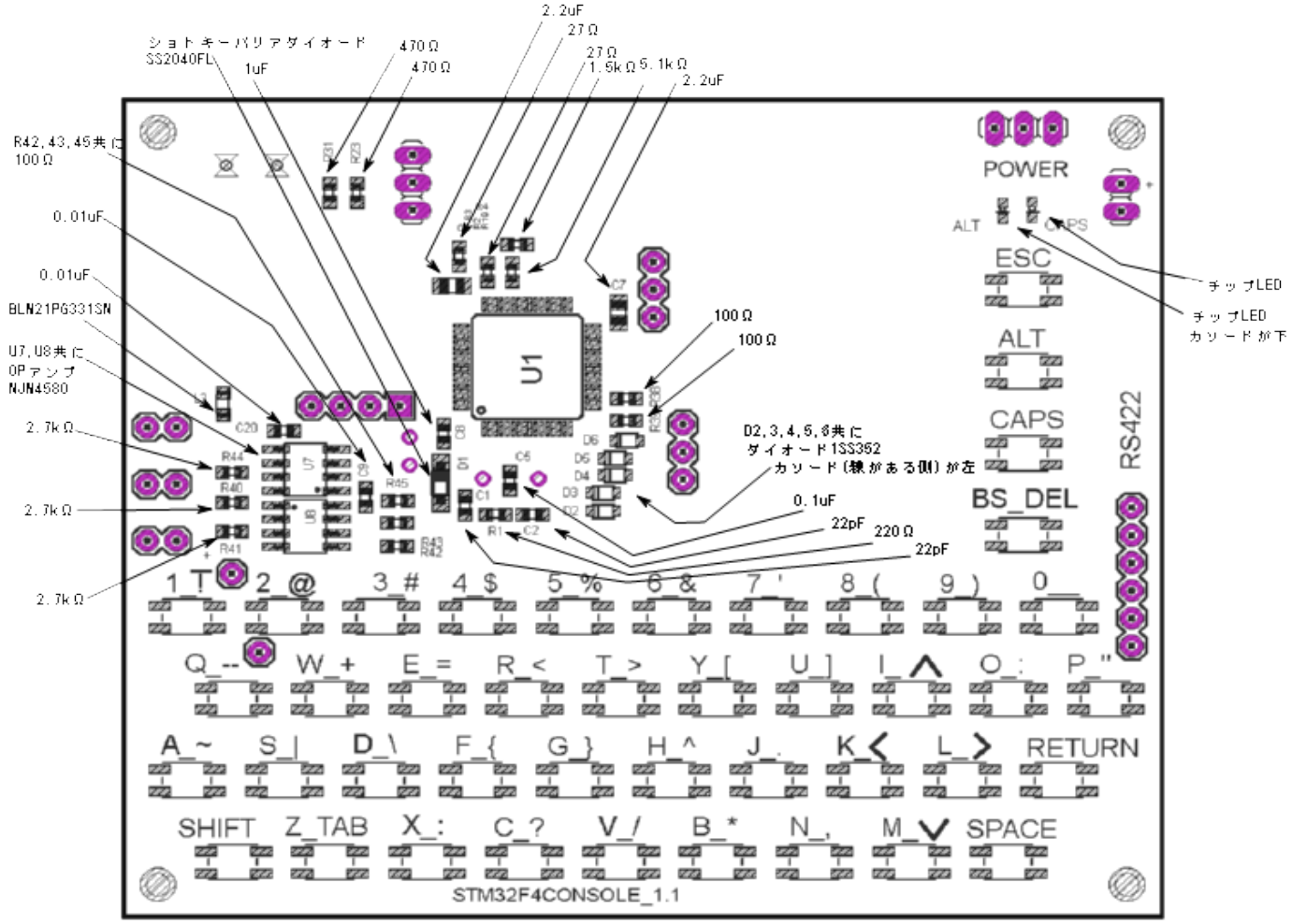
このマニュアルの最後に、板を切って曲げるタイプのケースの工作図面があります。組み立てには7〜8ミリ程度の長さのスペーサー4本が別途必要になります。

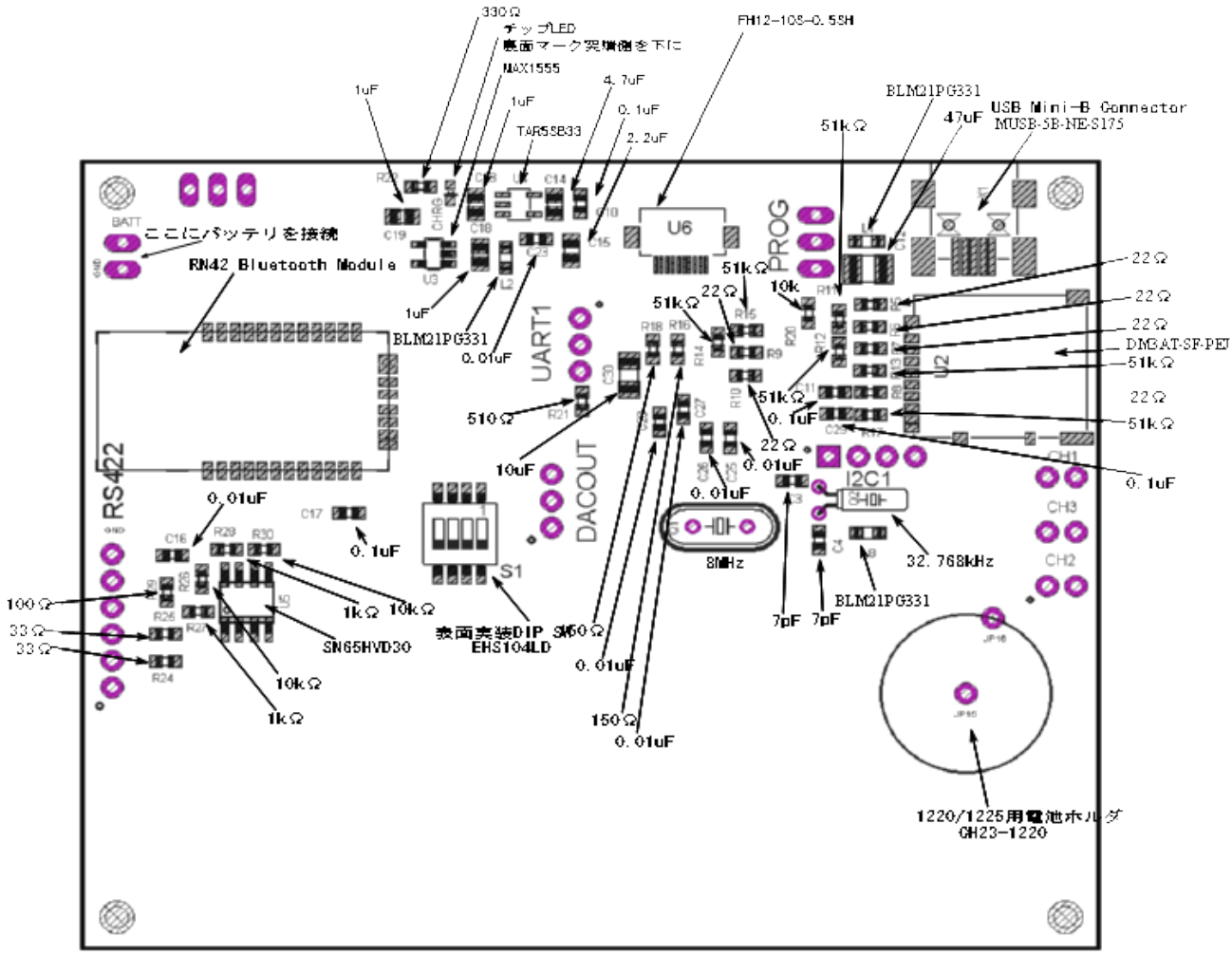
その他部品:

本基板は、裏面にBluetoothモジュールを半田付けして、Bluetoothを使えるようにすることができます。その他RS-422通信も部品追加で可能になります。Bluetoothモジュール RN42は、秋月電子で1500円で売っているものが使えます。

3:実装に関しての注意

部品を半田付けする位置を下に図示します。





4:プログラムの入手と書き込み

基板へ部品を実装したら、プログラムを書き込みましょう。 その為にはまずプログラムを書き込む環境を整えなければいけません。4.1 ではそれを説明します。
プログラムは用意されたものを書き込む方法と、自分でプログラムのソースコードとコンパイル環境を整え、自作する方法の 2 つがあります。まずは前者を試してください。これも 4.1 で説明しています。後者は 4.2、4.3 で説明します。自作することによって自由に機能を追加、改変することができるようになります。

4.1:プログラムの書き込み

Windows ユーザは以下の URL より書き込み用のツール DfuSe をダウンロード、インストールしてください。
STSW-STM32080:DfuSe USB device firmware upgrade STMicroelectronics extension
http://www.st.com/web/jp/catalog/tools/PF257916
書き込みに使うツールは DfuSe Demonstration です。
プレーヤ基板を USB ケーブルで PC と接続し、電源スイッチと Program スイッチを入れてください。ドライバのインストールが始めれば、マイコンとその周辺回路の半田づけはうまくいっています。ドライバのインストールが終わったら、DfuSe を起動します。
プレーヤ基板は OS から、STM Device in DFU mode として認識されます。DfuSe のウィンドゥ左上にも同様の表示がある筈です。次いで下部中央の Choose..ボタンを押して、書き込むファイルを選択します。
書き込むファイルは、以下の URL よりダウンロードしてください。

選択したら、DfuSe の画面、さっきのボタンのすぐ右、Upgrade のボタンを押してください。書き込みと、正しく書けたかを検査するベリファイが行われます。正常終了で、書き込みは成功、終了です。電源スイッチを落とし、Program スイッチを動作側にしてください。

Unix(linux,Mac)ユーザは、dfu-util を使います。

http://dfu-util.gnumonks.org/
サイトの指示に従って環境を構築してください。書き込むファイルは同じです。

4.2:プログラムソースの入手とコンパイル

プログラムのコンパイルには GCC コンパイラを使用します。Windows ユーザはまず Cygwin 開発環境を整えてください。Windows ユーザも Cygwin 環境を使うことで、以下の説明も Unix ユーザと同様に実行できます。

4.2.1:Cygwin の導入

Cygwin 導入には、
http://cygwin.com/
で Setup.exe をダウンロードし、実行することで好きなモジュールごとに選んでインストールすることになります。また Setup.exe を使うことで最新版へのアップデートも可能となります。
Setup.exe を実行し、[install from internet]を選択すると、インストールするディレクトルを聞かれます。ここはできるだけデフォルトの[C:\cygwin]のままで。
Install For は[All Users],Default Text file Type は[Unix]に。

インストールは、まず選択したパッケージファイルを指定したテンポラリディレクトリに展開してから行われます。Local Packege Directory はその指定です。
Select Your Internet Connection は[Direct connection]で。
Choose A Downloads site で、ダウンロード先を選択します。

Select Packages で、好きなパッケージを選んでインストールします。
Bin?は実行可能なバイナリ、Src?はソースコードを意味します。
この時点で、Cygwin システムとして必要最小限のパッケージが選択されています。
開発環境としてあと必要なのは、
+All
+Base <= Bin?全てを選択する
+Devel <= 以下のものの Bin?,Src?双方を選択する
gcc binutils flex bison libgmp3-dev libmpc-dev libmpfr-dev libncurses5-dev
+InterPreters <= gawk,m4,perl の Bin?を選択
あとは好きなものを。CVS,Subversion,git は選択べきです。

+Devel で選択する GCC は 4.*系列を選びましょう。ソースも忘れずに。

インストールが終了すると、デスクトップに Cygwin のアイコンが出来るのでダブルクリックして起動します。
見かけは DOS コマンドプロンプトに似ていますが、Unix 環境です。ためしに ls や cat を実行してみてください。

4.2.2:Newlib の導入

newlib は組み込み用の C/C++ライブラリです。
newlib はさっきの Cygwin のサイトからジャンプしたサイトからダウンロードできます。サイト左端のリンクリスト一番下の[sources.redhat.com]
http://sources.redhat.com/
再び左端の[Prejects]
http://sources.redhat.com/projects.html
下のほうの[newlib]
http://sources.redhat.com/newlib/ <=最初からここへ行くも良し
左端の[Download]から Source snapshot をとって来ます。一番下の文章中の[newlib ftp directory]というリンクの先から、一番新しいものをダウンロードしてきます。
*もちろん、CVS で落とすのも OK
ダウンロードしたファイルは C:/Cygwin/usr/src にコピーします。
解凍は Cygwin 環境で行います。まず cd /usr/src で、ダウンロードしたファイルの位置に飛びます。Cygwin の中では、C:\Cygwin が / つまりルートディレクトリとなります。
tar -zxvf newlib-2.1.0.tar.gz
これで解凍されます。

4.2.3:binutils の準備

/usr/src には、ソースコードのダウンロードを選択したパッケージが、既に展開されていると思います。
その中に cd し、次のようなコマンドを実行します
./configure --target=arm-m4-eabi --prefix=/opt/stm32-tools/ --enable-interwork --enable-multilib --disable-nls --disable-libssp

configure コマンドは環境に応じてコンパイルの下準備をします。
-target オプションは ARM アーキテクチャの elf バイナリ版を作るということを意味しています。elf についてはここでは触れません。
-prefix オプションは、クロスコンパイル環境をどこに構築するかを示します。ここでは/usr/cross を推奨します。先にディレクトリを作っておきましょう。
mkdir /usr/cross
-disable-nls は多言語対応オプションを外すことを意味しています。
次に、
make
を実行します。無事終了したら(しばらくかかります)、
make install
を実行します。

4.2.4:GCC の準備

/usr/src/gcc-4.*.* へ cd します。
まず、newlib へのシンボリックリンクを張ります。まぁ、ショートカットみたいなもんです。
ln -s ../newlib-2.1.0/newlib .

コンパイル先ディレクトリを掘ります。

mkdir dist
cd dist
コンフィギュレーションを行います。
../configure --target=arm-m4-eabi --prefix=/opt/stm32-tools/ --enable-interwork --enable-multilib -with-gmp=/usr/local -with-mpfr=/usr/local -with-mpc=/uar/local --enable-languages="c" --with-newlib --with-headers=../../newlib-2.1.0/newlib/libc/include/ --disable-libssp --disable-nls
そして、
make CFLAGS_FOR_TARGET="-mcpu=cortex-m4 -mthumb -mfloat-abi=softfp -mfpu=fpv4-sp-d16"
make install
dist ディレクトリの中身を消去(make clean ではうまくいかない場合があった)し、再度コンフィギュレーションを行います。
../configure --target=arm-m4-eabi --prefix=/opt/stm32-tools/ --enable-interwork --enable-multilib -with-gmp=/usr/local -with-mpfr=/usr/local -with-mpc=/uar/local –enable-languages="c,c++" --with-newlib --with-headers=../../newlib-2.1.0/newlib/libc/include/ --disable-libssp --disable-nls
make CFLAGS_FOR_TARGET="-mcpu=cortex-m4 -mthumb -mfloat-abi=softfp -mfpu=fpv4-sp-d16" CXXFLAGS="-mcpu=cortex-m4 -mthumb -mfloat-abi=softfp -mfpu=fpv4-sp-d16" all
make install
これで GCC が C++まで使用可能となります。

4.2.2:プログラムソースの準備

バージョン管理ツール Git を使い、Github サイトから以下のようなコマンドでプログラムソースを取得してください。

git clone https://github.com/mizuki_tohru/stm32f4-console stm32f4-console

以後もデバッグ、アップデートの度にプログラムソースは更新されますので、最新版(や過去のバージョン)を Git を使って取得してください。

4.2.3:必要なライブラリソースの準備

以下のようなライブラリが必要になります。

STM32F4 のライブラリ

STSW-STM32065 STM32F4 DSP and standard peripherals library を以下の URL より入手してください。
http://www.st.com/web/catalog/tools/PF257901

STM32F105/7, STM32F2 and STM32F4 USB on-the-go Host and device library (UM1021) から USB 関連ライブラリを選りだして上のライブラリにコピーしてください。
http://www.st.com/web/catalog/tools/PF257882

東雲フォントと NKF ライブラリ(漢字を使う場合)
http://openlab.jp/efont/dist/shinonome/
から、shinonome-0.9.11p1.tar.bz2 を入手して makefile の示す位置に展開しておいてください。
漢字を使用する場合 libnkf により文字コード対応を行いますが、libnkf-1.0.0.tar.gz の配布が中止されたようなので、他の libnkf 実装を試してみてください。

fatff 他ライブラリについて、ねむいさんのブログより以下のライブラリを入手してください。
TFT/OLED Control Sample with ChaN's FatFs(SDIO&MMC Driver)
http://nemuisan.blog.bai.ne.jp/?eid=192848

4.2.4:プログラムのコンパイル

プログラムソースの位置はは gcc やライブラリにパスが通っている必要があります。パスが通っていなければ、makefile で直接指定してください。
取得したライブラリのバージョンが makefile に書かれているものと食い違っている場合があります。実際に取得したライブラリのバージョンにあわせて makefile は書き換えて下さい。
コンパイルに成功すると様々なファイルが生成されますが、使用するのは main.s19 です。これを DfuSe ツールの中の DFU File manager で拡張子 dfu のファイルに変換し、4.1 で説明した手順でこれをボード上のマイコンに書き込みます。

5:動作について

右上 POWER スイッチは、バッテリーからの給電と USB からの給電、この 2 つの給電方式を切り替えるスイッチです。右がバッテリーから、左が USB からになります。
使い初めや充電が必要な場合は、まず POWER スイッチを左にスライドさせて USB-MiniB コネクタを接続してください。基板裏で赤い LED が光ります。これが消えると充電完了です。そのまま USB コネクタを引き抜いてください。
USB ケーブルが接続していない状態で、POWER スイッチを右にスライドさせると動作開始です。てのりマイコンはマイクロ SD カードが刺さっていればこれを検査した後、BASIC インタプリタを起動します。OK と表示されて、キー操作ができるようになります。
キーボードはシンクレア ZX スペクトラムのものを一部アレンジしたものになっています。右上の 2 つの LED が両方とも消灯しているときは、キーボードは印刷左側、数字とアルファベット、但し小文字を出力します。SHIFT キーを押しながら他のキーを押すと大文字が出力されます。また、CAPS キーを押すと LED 右側が点灯し、大文字が打てるようになります。更にもう一度 CAPS キーを押すと LED は消灯し、小文字に戻ります。
ALT キーを押すと、左側の LED が点灯し、キーボードの印刷右側が打てるようになります。更にもう一度押すと LED は消灯し、キーは元の数字とアルファベットの配列に戻ります。ALT キーと CAPS キーが双方とも押された場合、ALT キーのほうが優先されます。
BS_DEL キーはアルファベットが小文字を打てるときはバックスペース、大文字を打てるときはデリートキーとして働きます。このキーは ALT キーを無視します。
ESC キーを押すと、BASIC インタプリタを終了し、シリアルモニタに動作を切り替えます。更に押すとアナログモニタに動作は切り替わります。更に押すと日付時刻設定モードになりますので、事前に 1220 または 1225 ボタン電池を基板裏の電池ホルダに入れてから、日付時刻の設定をおこなってください。
日付時刻設定モードから更に ESC キーを押すと、BASIC インタプリタに動作は再び戻ります。

日付設定の仕方
カーソルキーの左右(K_<,L_>)でカーソルを移動して、カーソルの上下(I_↑,M_↓)で値を変更します。値を確定したら、RETURN キーを押してください。それで値が設定されます。

シリアルモニタ
プログラムはデフォルトでは UART1 の入出力をサポートしています。Bluetooth モジュールを使う際は、UART6 を替わりに有効化してください。
受信内容はリアルタイムで表示され、キーで入力された内容はリターンキーで送信されます。初期値は 115200bps ｍ 8 ビット、ノンパリ、ストップビット 1 です。

アナログモニタ
液晶左下付近に 2 ピンのスルーホールが 3 つ並んでいますが、ここに端子を接続して外部電圧を測定、表示することができます。表示されるのは(今のところ)ch1 と ch2 の二つです。スルーホールと ch の関連は基板裏側のシルクを見て確認してください。
電圧は 3.3V まで入力できます。(入力する OP アンプにはレールツーレールタイプを使っています。)ピンは外側が GND、内側が電圧入力になります。

BASIC インタプリタ

BASIC インタプリタは uBASIC(dunkels.com/adam/ubasic)に拡張を加え、エディタを追加したものです。横 50 文字、256 行まで(メモリには 12kbyte まで)の記述ができます。

変数には 16 ビット整数及び 32 ビット整数、文字列型と配列が使用できますが、使用できる数と名前には制限があります。変数は全てアルファベット小文字 a から z、大文字 A から N までの 40 種のみとなります。種別は以下の通りです。

a-z	整数型 16 ビット符号付き整数。
A,B,C,D	配列型 16 ビット整数を最大長 2048 個まで保持できる。
E,F,G,H	文字列型 8 ビット整数を最大長 4096 個まで保持できる。
I,J,K,L,M,N	大整数型 32 ビット符号付き整数。

変数は使用前に LET 文で宣言をおこなわなければいけません。行わなかった場合動作は不定です。

let 文

構文: let 変数名

例:

```
100 let a
120 a = 100
```

BASIC の各行には行番号が必要です。実行は行番号の若いものから順におこなわれます。行番号の無い行はリターンキーでその行は即時実行されます。

制御構文として if-then、for-to-next、goto、gosub、return、call、end があります。

if-then 文

構文: if 条件 then 処理

例:

```
100 if a > 10 then goto 200
110 print "little"
```

for-to-next 文

構文: for ループ変数初期化 to ループ終了値 値増加は 1 づつ
　　ループされる処理
　　next ループ変数

例:

```
100 for a = 0 to 100
110 print a
120 next a
```

goto 文

行番号の位置まで飛ぶ。

例:

```
100 goto 200
```

gosub-return 文

行番号の位置まで飛び、実行中に return に遭遇したら gosub の位置まで戻ってくる。

例:

```
100 gosub 200
110 print "second"
120 end
200 print "first"
210 return
```

end 文

BASIC プログラムの動作を終了して、エディタの入力画面に戻ります。

例:

```
100 print "end"
110 end
```

run 命令

run は行番号なし行でしか実行されない。run 文が実行されると、行番号つき BASIC の各行が一番若い行から順に実行される。

例:

```
run
```

list 命令

list は行番号なし行でしか実行されない。list 文が実行されると、行番号つき BASIC の各行が一番若い行から順に出力される。

例:

```
list
```

files 命令

files は行番号なし行でしか実行されない。files 文が実行されると、SD カード内のファイルの一覧が表示される。

例:

```
files
```

rem 文

rem はコメント文で、この文から行末までインタプリタには実行されない。

例:

```
100 print "hoge" rem hogehogehogehoge
```

pset 文
pset 文は画面に点を打つ。
構文 pset x y 1 (1 で黒点、0 で白点) x は 0 から 399 まで、y は 0 から 239 まで。左上が画面原点。
例:
```
100 let x
110 x = 100
120 pset x 100 1
```

refresh 文
refresh は画面を更新する。pset 文は画面に転送する前のイメージバッファを更新するが、画面への転送をそのタイミングでは行わない。refresh 文は強制的にイメージバッファを画面へと転送する。
例:
```
100 refresh
```

cls 文
cls 文は画面を消去する。
例:
```
100 cls
```
例:
```
cls
```

load 文
load 文は SD カードから指定のファイルをロードする。拡張子が BAS のものは内容をプログラムとしてロードする。ファイル名はダブルクォーテーションで囲わなければならない。
拡張子が TXT のものは内容を文字列型変数 H に読み込む。但し 4096 バイトまでしか読み込むことはできない。
拡張子が DAT のものは内容を配列型変数 D に読み込む。このときファイル内容をバイナリデータとみなして先頭から 4 キロバイトまでを読み込む。
拡張子が BMP のものは Windows ビットマップ 2 値イメージファイルとみなして読み込み後画面表示をおこなう。この際先に表示されていた内容はすべて消去される。ファイルは 400x240 ビットのモノクロ二値データでなければならない。
例:
```
load "hoge.bas"
```

save 文
save 文は SD カードへ指定のファイル名で書き込む。拡張子が BAS のものはエディタで編集中の行番号付プログラムが格納される。同名のファイルが存在した場合は上書きされる。
拡張子が TXT のものは、文字列型変数 H の内容を文字列終端もしくは 4096 バイトまで指定のファイルに書き込む。同名のファイルが存在した場合は上書きされる。
拡張子が DAT のものは、配列型変数 D の内容を終端もしくは 4096 バイトまで指定のファイルに書き込む。同名のファイルが存在した場合は上書きされる。
拡張子が BMP のものは、現在のスクリーンショットを指定のファイルに書き出す。同名のファイルが存在した場合は上書きされる。
例:
```
save "hogehoge.bas"
```

wait 文
wait 文は指定ミリ秒間動作を停止する。
例:
```
100  wait 100
```

input 文
input 文は キーボードから入力を受け取る文である。
代入変数は文字列型、改行コードが出るまで、または変数が一杯になるまで動作する。
例:
```
100 let E
120 E = input
```

inp 文
inp 文は キーボードから入力を一文字だけ受け取る文である。キーボードからひと文字受け取るまで入力待ちをする。
例:
```
100 let a
110 a = inp
```

inr 文
inr 文は キーボードから入力を受け取る文である。入力待ちを行わず、呼び出すと直近のキーボード入力ひと文字を返す。
例:
```
100 let a
110 a = inr
```

ina 文
ina 文は ADC アナログ入力を受け取る文である。
構文 a = ina 1 (=チャンネル番号 0,1,2)
例:
```
100 let a
110 a = ina 1
```

print 文
print 文は指定の文字列を出力する。文字列型変数も出力できる。
例:
```
 100 print "good hoge\n"
 110 print E
```

演算子として +,-,*,/,%,&,|,>,<,=,がある。結合制御に()が使用できる。
演算子に優先順位は無い。

入出力端子のピン定義は以下のようになっています。

UARTOUT: CMOS レベルシリアルポート

Pin1	USART1_TX
Pin2	USART1_RX
Pin3	GND

DACOUT:DAC 出力

Pin1	DAC1
Pin2	DAC2
Pin3	GND

I2C1:I2C ポート

Pin1	VCC3.3V
Pin2	I2C1_SCL
Pin3	I2C1_SDA
Pin4	GND

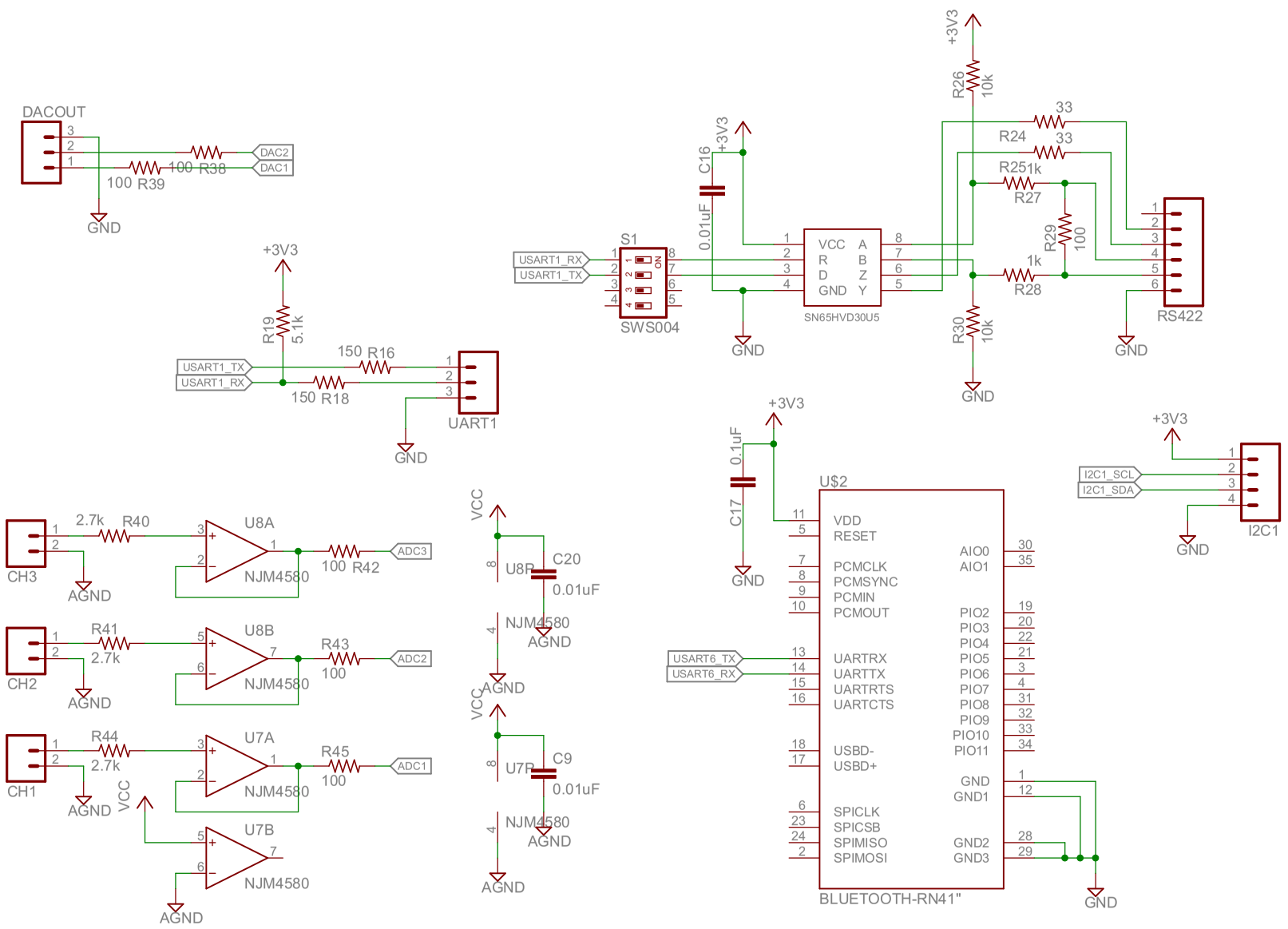
RS422: RS-244 レベルシリアルポート

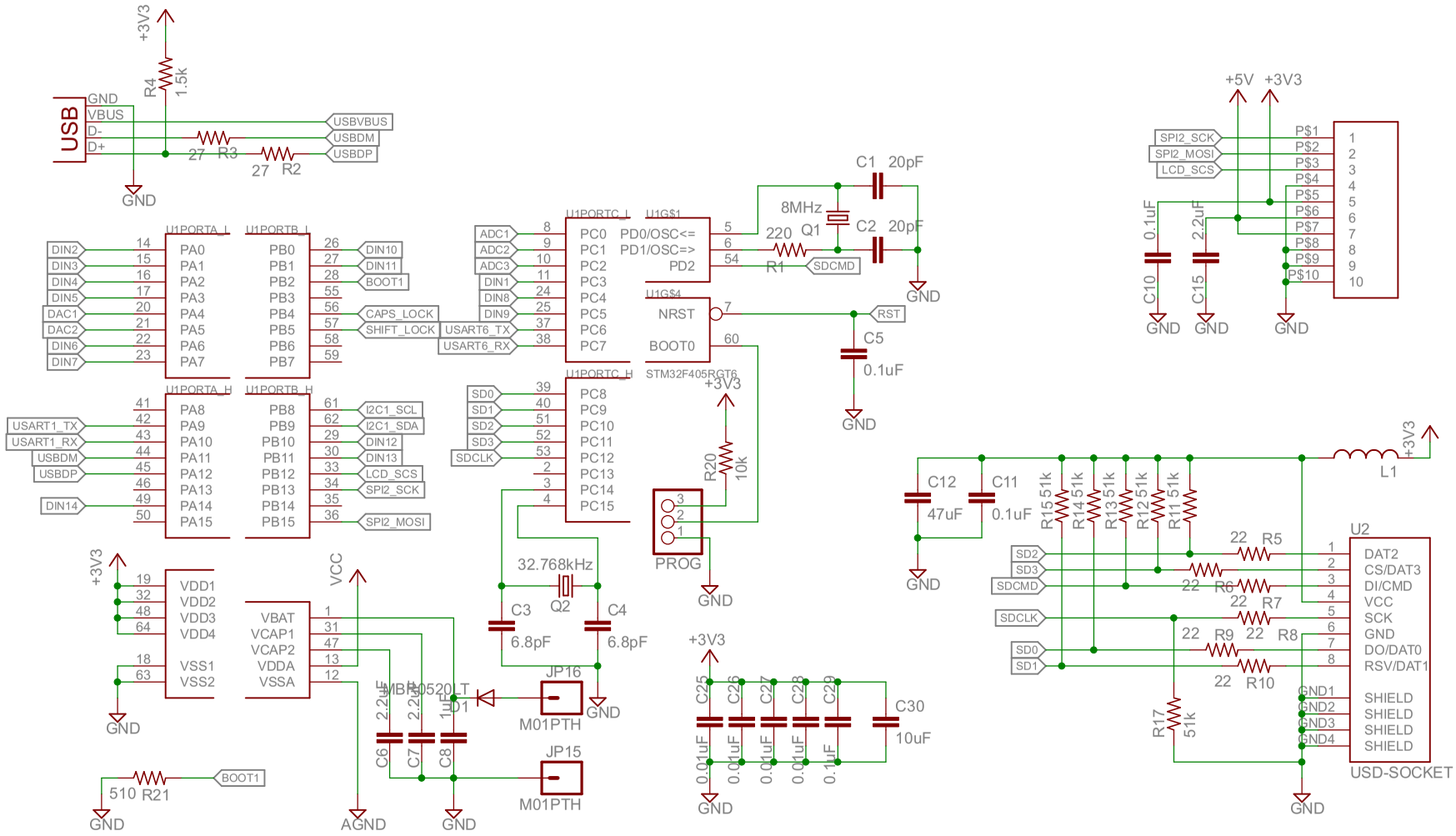
Pin1	NC
Pin2	USART1 TX +
Pin3	USART1 TX-
Pin4	USART1 RX+
Pin5	USART1 RX-
Pin6	GND

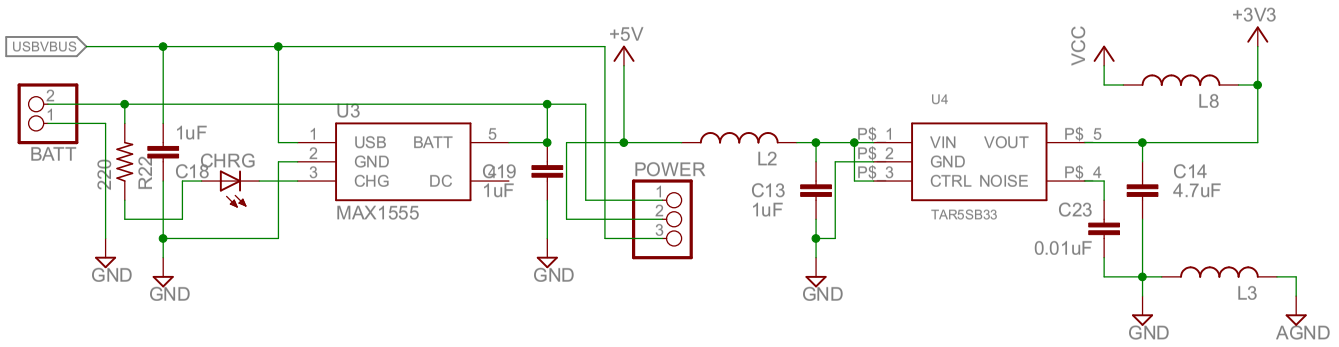
CH1,CH2,CH3 アナログ入力

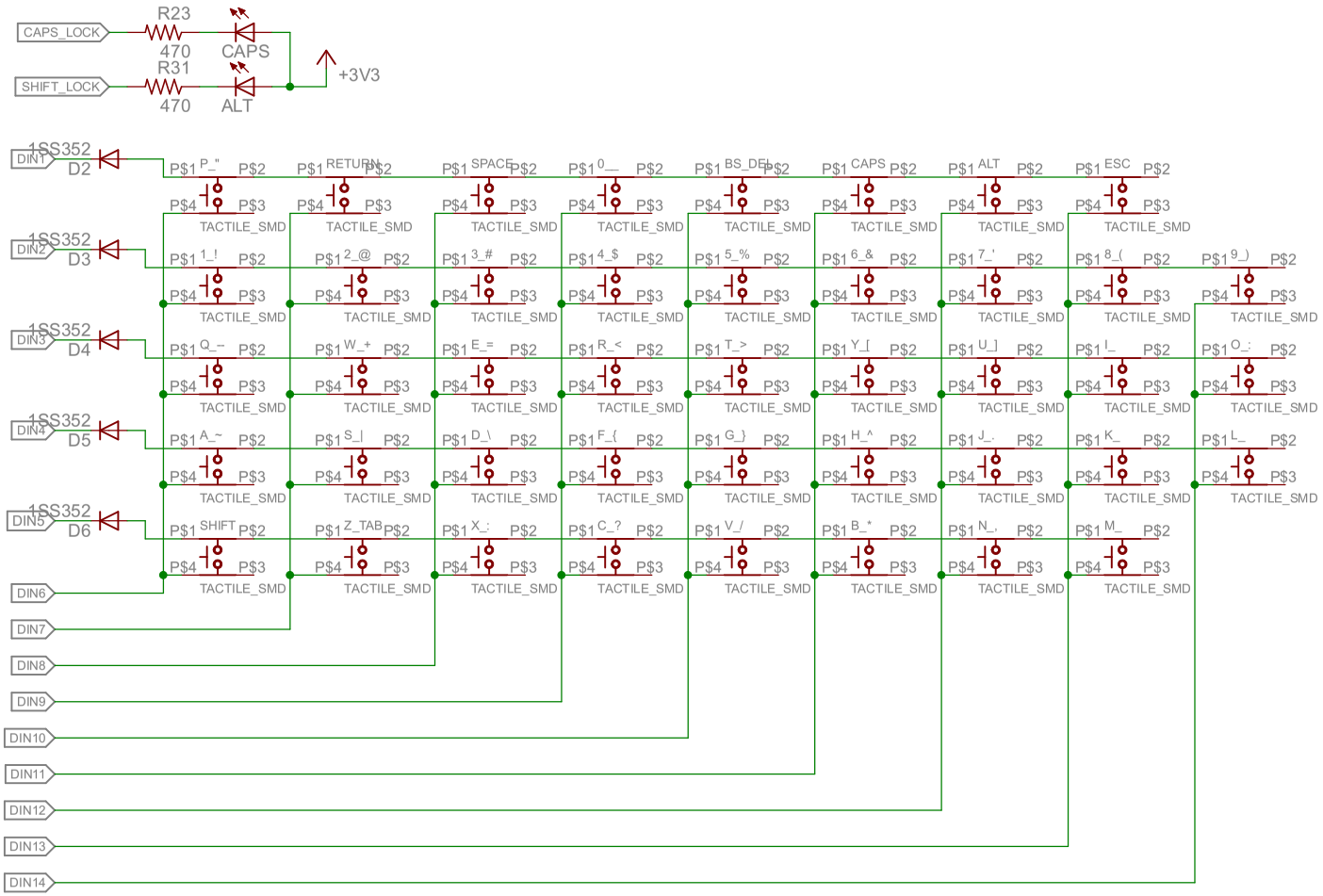
CH1		CH2		CH3	
Pin1	ADC3	Pin1	ADC2	Pin1	ADC1
Pin2	GND	Pin2	GND	Pin2	GND

6:回路図

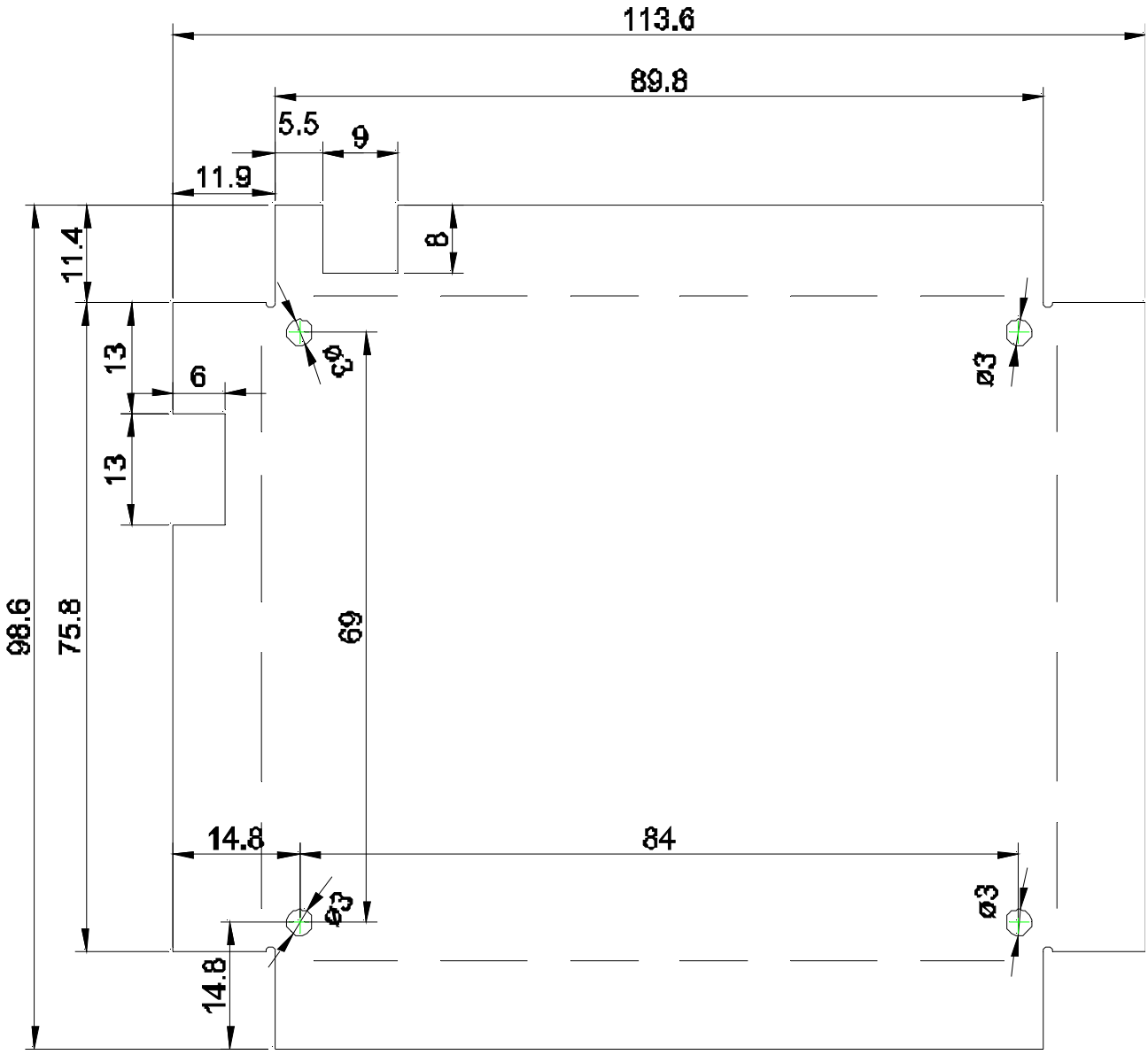

DACOUT
DAC2
DAC1
100 R38
100 R39
GND
+3V3
R19
5.1k
USART1_TX
USART1_RX
150 R16
150 R18
UART1
GND
S1
SWS004
USART1_RX
USART1_TX
C16
0.01uF
+3V3
GND
VCC
R
D
GND
A
B
Z
Y
SN65HVD30U5
R26
10k
33
R24
33
R25
1k
R27
R29
100
1k
R28
R30
10k
GND
RS422
CH3
2.7k
R40
U8A
NJM4580
100 R42
ADC3
AGND
CH2
R41
2.7k
U8B
NJM4580
R43
100
ADC2
AGND
CH1
R44
2.7k
U7A
NJM4580
R45
100
ADC1
AGND
VCC
U7B
NJM4580
AGND
VCC
U8B
C20
0.01uF
NJM4580
AGND
VCC
U7B
C9
0.01uF
NJM4580
AGND
+3V3
C17
0.1uF
GND
U$2
VDD
RESET
PCMCLK
PCMSYNC
PCMIN
PCMOUT
UARTRX
UARTTX
UARTRTS
UARTCTS
USBD-
USBD+
SPICLK
SPICSB
SPIMISO
SPIMOSI
AIO0
AIO1
PIO2
PIO3
PIO4
PIO5
PIO6
PIO7
PIO8
PIO9
PIO10
PIO11
GND
GND1
GND2
GND3
USART6_TX
USART6_RX
BLUETOOTH-RN41"
GND
+3V3
I2C1_SCL
I2C1_SDA
I2C1
GND


+3V3
R4
1.5k
USB
GND
VBUS
D-
D+
USBVBUS
USBDM
USBDP
27 R3
27 R2
GND
U1PORTA_L
U1PORTB_L
DIN2
DIN3
DIN4
DIN5
DAC1
DAC2
DIN6
DIN7
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
DIN10
DIN11
BOOT1
CAPS_LOCK
SHIFT_LOCK
U1PORTC_L
ADC1
ADC2
ADC3
DIN1
DIN8
DIN9
USART6_TX
USART6_RX
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
U1G$1
PD0/OSC<=
PD1/OSC=>
PD2
8MHz
C1 20pF
220 Q1
C2 20pF
R1
SDCMD
GND
U1G$4
NRST
BOOT0
RST
C5
0.1uF
GND
U1PORTA_H
U1PORTB_H
USART1_TX
USART1_RX
USBDM
USBDP
DIN14
PA8
PA9
PA10
PA11
PA12
PA13
PA14
PA15
PB8
PB9
PB10
PB11
PB12
PB13
PB14
PB15
I2C1_SCL
I2C1_SDA
DIN12
DIN13
LCD_SCS
SPI2_SCK
SPI2_MOSI
U1PORTC_H
STM32F405RGT6
SD0
SD1
SD2
SD3
SDCLK
PC8
PC9
PC10
PC11
PC12
PC13
PC14
PC15
+3V3
R20
10k
PROG
GND
32.768kHz
Q2
C3
6.8pF
C4
6.8pF
JP16
M01PTH
GND
+3V3
VDD1
VDD2
VDD3
VDD4
VSS1
VSS2
VBAT
VCAP1
VCAP2
VDDA
VSSA
VCC
MBR0520LT
D1
C6 2.2uF
C7 2.2uF
C8 1uF
JP15
M01PTH
AGND
GND
BOOT1
510 R21
GND
+3V3
C25 0.01uF
C26 0.01uF
C27 0.01uF
C28 0.01uF
C29 0.1uF
C30
10uF
GND
+5V +3V3
SPI2_SCK
SPI2_MOSI
LCD_SCS
P$1
P$2
P$3
P$4
P$5
P$6
P$7
P$8
P$9
P$10
C10 0.1uF
C15 2.2uF
GND GND GND
C12
47uF
C11
0.1uF
GND
+3V3
L1
R15 51k
R14 51k
R13 51k
R12 51k
R11 51k
SD2
SD3
SDCMD
SDCLK
SD0
SD1
22 R5
22 R6
22 R7
22 R9
22 R8
22 R10
R17
51k
U2
DAT2
CS/DAT3
DI/CMD
VCC
SCK
GND
DO/DAT0
RSV/DAT1
GND1
GND2
GND3
GND4
SHIELD
SHIELD
SHIELD
SHIELD
USD-SOCKET
GND


USBVBUS
BATT
2
1
220
R22
1uF
C18
CHRG
GND
GND
U3
USB
GND
CHG
BATT
DC
1
2
3
5
4
MAX1555
C19
1uF
GND
+5V
POWER
1
2
3
L2
C13
1uF
GND
U4
P$ 1
P$ 2
P$ 3
VIN
GND
CTRL
VOUT
NOISE
TAR5SB33
P$ 5
P$ 4
C23
0.01uF
C14
4.7uF
GND
VCC
+3V3
L8
L3
AGND


CAPS_LOCK
R23
470
CAPS
SHIFT_LOCK
R31
470
ALT
+3V3
DIN1
1SS352
D2
DIN2
1SS352
D3
DIN3
1SS352
D4
DIN4
1SS352
D5
DIN5
1SS352
D6
DIN6
DIN7
DIN8
DIN9
DIN10
DIN11
DIN12
DIN13
DIN14
P_"
RETURN
SPACE
0__
BS_DEL
CAPS
ALT
ESC
1_!
2_@
3_#
4_$
5_%
6_&
7_'
8_(
9_)
Q_~
W_+
E_=
R_<
T_>
Y_[
U_]
I_
O_:
A_~
S_|
D_\
F_{
G_}
H_^
J_.
K_
L_
SHIFT
Z_TAB
X_:
C_?
V_/
B_*
N_,
M_
TACTILE_SMD

この図面は、板材を曲げて製作するタイプのケースの図面例です。



2014 年 8 月 17 日 初版第一刷
航天機構