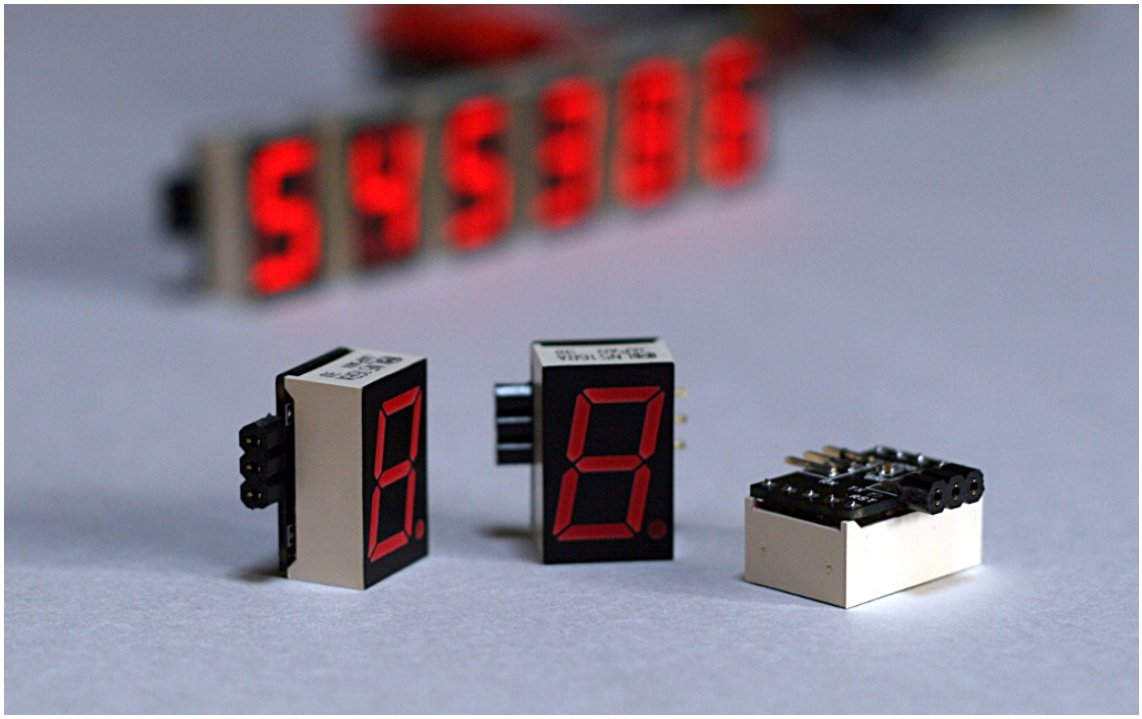
マイコン搭載 7 セグメント LED モジュール

7 セグブロック

Rev 1.1



■ 概要

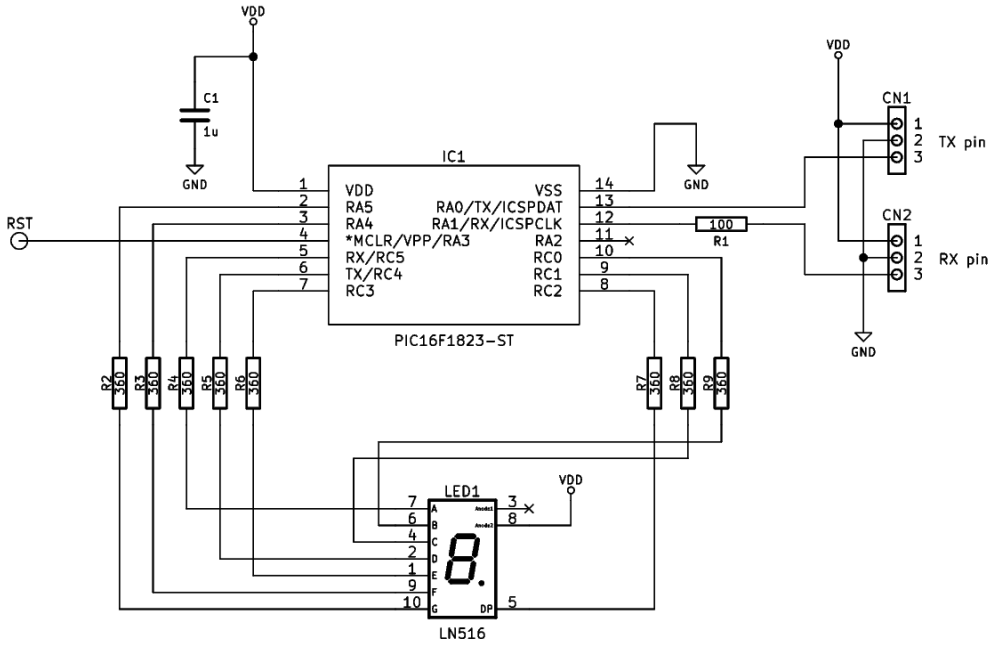
- 7 セグブロックは、7 セグメント LED に、マイコンを搭載した基板を取り付け、一体化したモジュールです。UART 経由で表示を制御できます。
- 7 セグブロックを連結させて表示桁数を任意に増やすことができます。接続できる桁数に制限はありません。(実際には通信速度および電源に依存します)
- ファームウェアのブートローダ機能により、ユーザ自身が作成したプログラムを実行することが可能です。

■ 仕様

動作電圧: 2.5V ～ 5.5V (推奨電圧 5.0V)
マイコン: PIC16F1823 I/ST (TSSOP 14pin)
動作周波数: 16MHz (*1)
7 セグメント LED : LN516RA
通信方式: 調歩同期方式 9600bps / パリティ無 / ストップビット 1bit (*1)

(*1) ユーザプログラムにより変更可能

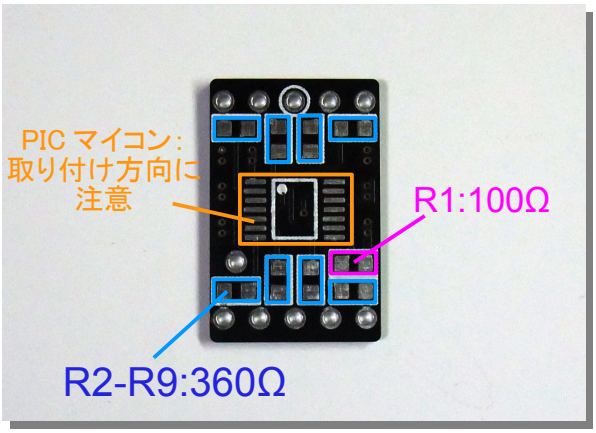
■ 回路図



■ 部品表

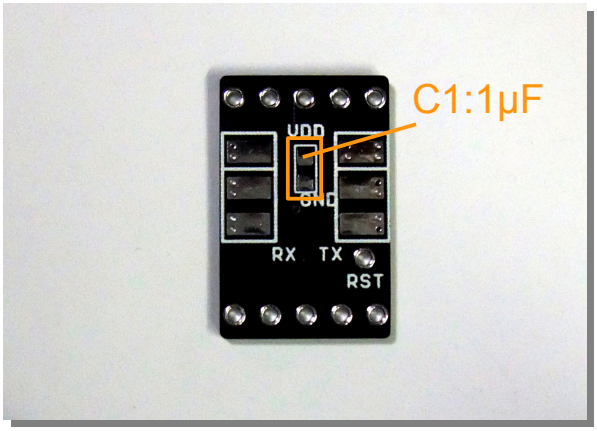
部品番号	型番	個数
U1	PIC16F1823 I/ST (TSSOP 14pin)	1
R1	100Ω (1/10W, 1608)	1
R2 - R9	360Ω (1/10W, 1608)	8
C1	1μF (25V, 1608)	1
LED1	LN516RA	1
CN1	ロープロファイルピンソケット	1
CN2	ロープロファイルピンヘッダ (7.7mm)	1

■ 組み立て手順

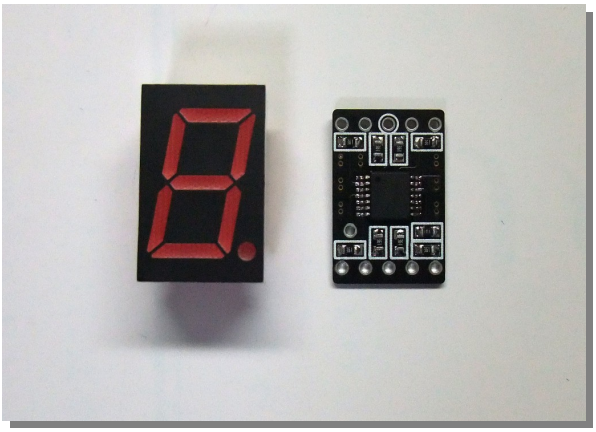


1. マイコンと抵抗の半田付け

マイコンは部品の向きを十分に確認して半田付けしてください。また R1 のみ抵抗値が異なるので注意してください。

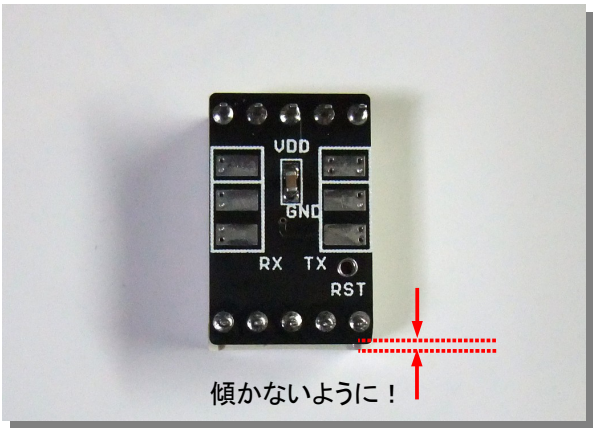


2. 基板裏面、コンデンサの半田付け



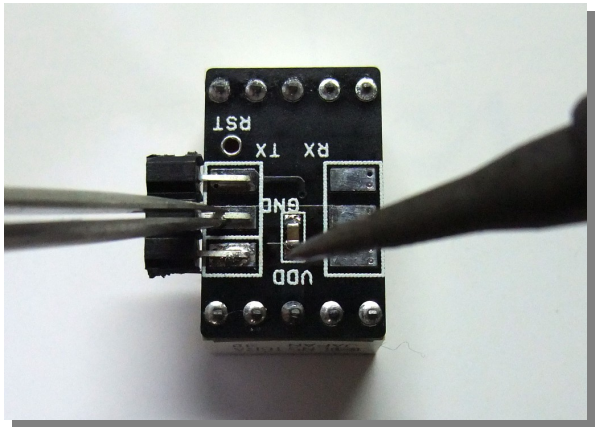
3. 7 セグメント LED の半田付け

基板に対する 7 セグメント LED の向きは左の写真の通りです。（このように並べた状態から 7 セグメント LED を並行移動させ基板に被せる）



7 セグメント LED を、基板に対して傾きがないように確認しながら半田付けをしてください。

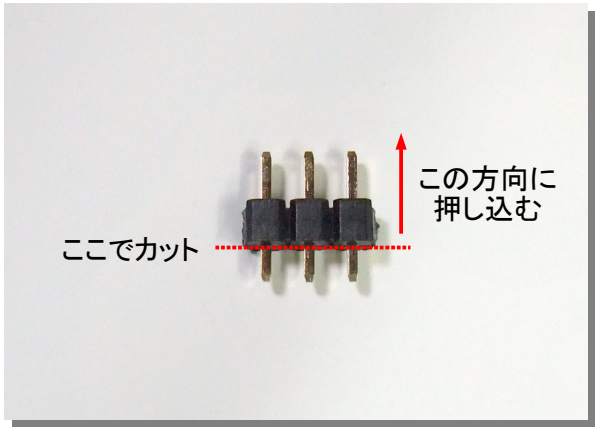
ピンヘッダやピンソケットを取り付けず、独自の方法で 7 セグブロックを使用する場合は以降の工程は不要です。



4. TX 側ピンソケットの半田付け

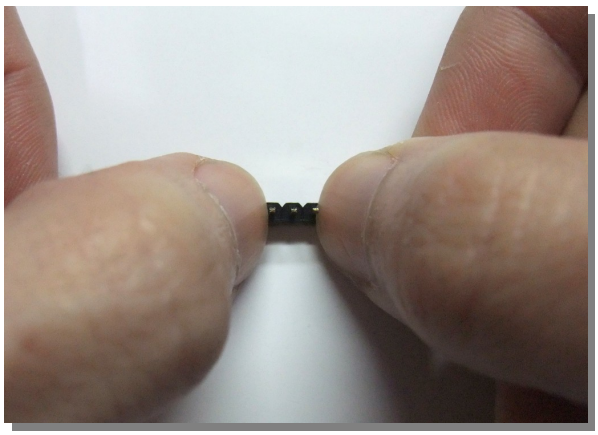
あらかじめ半田ごてに半田を溶かしておきます。ピンソケットの位置を調整し、リード部分をランドに押さえつけながら、1 つのランドに半田を流し込みます。

傾きがないことを確認できたら、残り全ての端子をハンダ付けします。

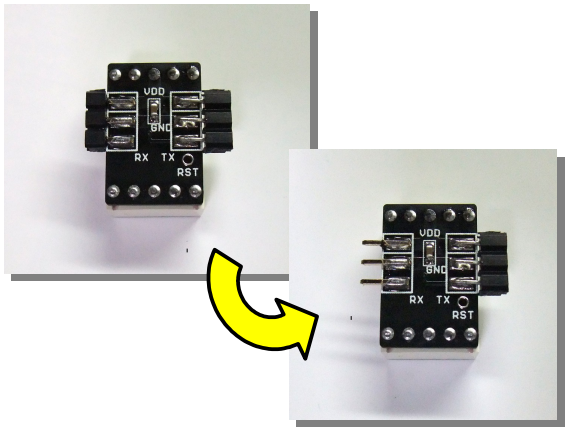


5. RX 側ピンヘッダの加工

初めにピンヘッダの短い側をニッパで切断します。次にピンヘッダのインシュレータ（黒い部分）を切断した側と反対側に押し下げます。



インシュレータの両端を親指で押さえて、ゆっくりと押し込むようにするとうまくできます。



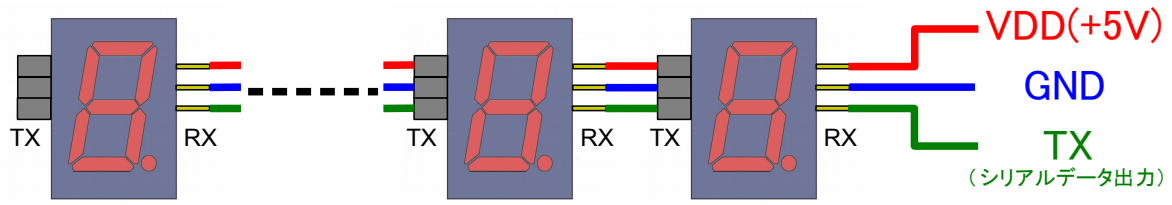
6. RX 側ピンヘッダの半田付け

TX 側と同様の手順で取り付けます。半田付け後はインシュレータを取り除きます。

インシュレータ強引に取り除こうとすると、銅箔が剥離する場合があるので、力をゆっくり加えながら押し出してください。

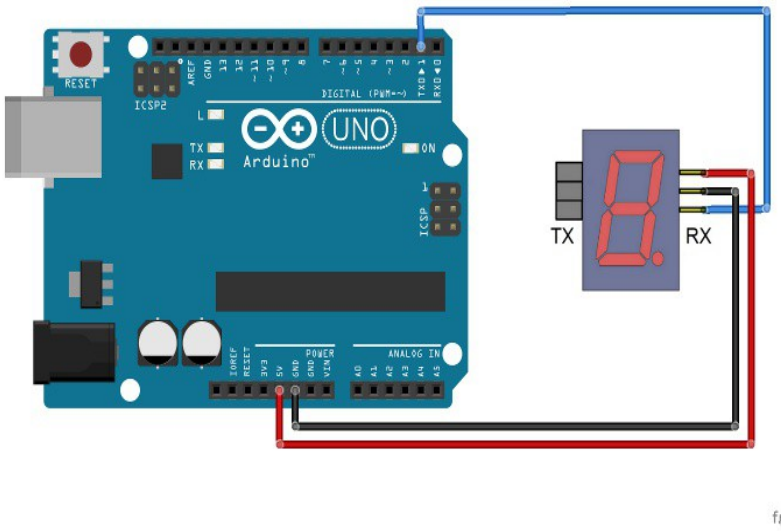
■ 使用方法

7 セグブロックは下図のように接続して使用します。通信速度と電源容量が許す限り、何個でも接続することができます。



例えば、Arduino に接続して使用する場合、初めに 7 セグブロックの電源端子と Arduino の電源端子（5V/GND）をそれぞれ接続します。そして 7 セグブロックの RX 端子と Arduino の TX 端子（デジタル端子 1 番ピン）を接続します。

電源電圧がシリアル信号の振幅より低くならないように注意してください。



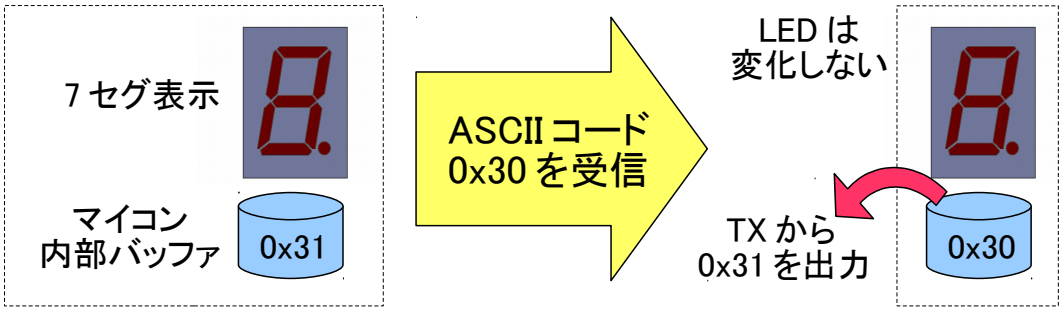
下記ような Arduino スケッチで簡単に表示させることができます。動作の詳細は次のページより解説します。

```
unsigned int num = 8;

void setup() {
  Serial.begin(9600);
}

void loop() {
  Serial.println(num, DEC);
}
```

■ 動作の解説

電源投入後、7 セグブロックは ASCII コード受信して 16 進表示をするモード（ノーマルモード）に入ります。ノーマルモードの詳細は次の通りです。

1. ASCII コードを受信した場合、そのデータを内部のバッファに保存すると同時に、すでにあるデータを TX 端子から送信します。このとき LED の表示は変化しません。



ここでの ASCII コードとは、

DELIMITER (0x0D)

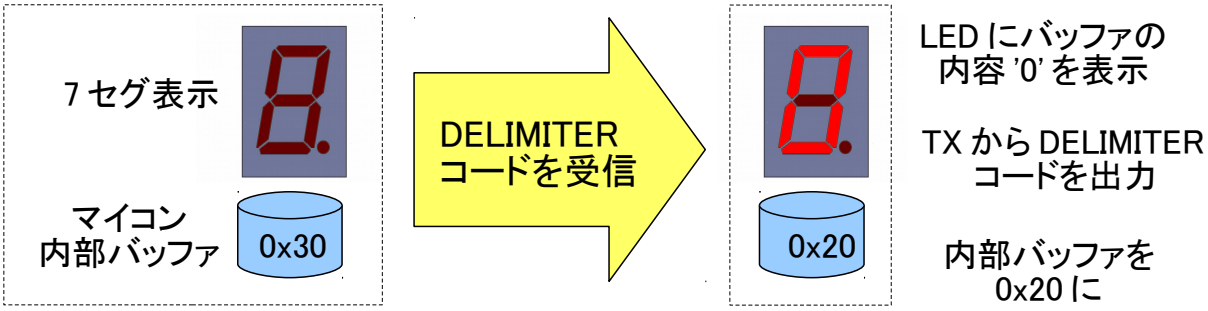
NEWLINE (0x5C)

ユーザプログラムモードへ移行する際の先頭コード（0xaa）

以外のコードになります。

2. DELIMITER コード (0x0D: 'CR') を受信した場合、バッファにあるデータを表示すると同時に、TX 端子から DELIMITER コードを送信します。

さらに、バッファの内容を 0x20（スペースのコード）に置き換えます。



7 セグ LED に表示できるのは 16 進数表記のみです。

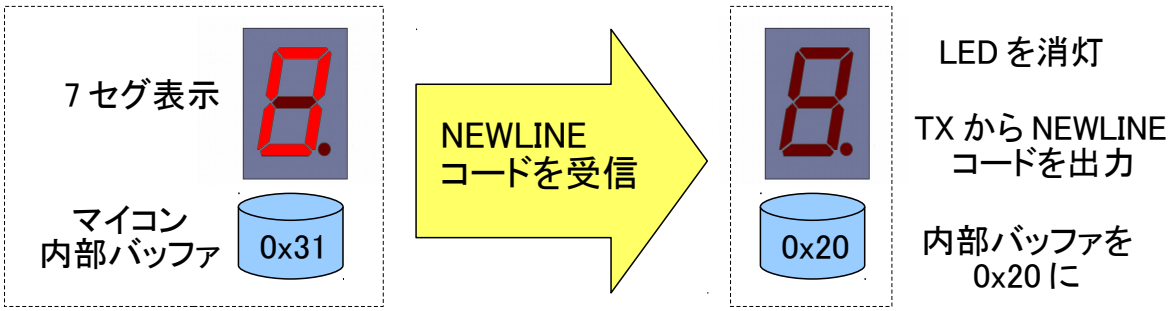
ASCII コードの 0x30('0') ～ 0x39('9'), 0x41('A') ～ 0x46('F'), 0x61('a') ～ 0x66('f') に該当します。

また、7 セグのドットを表示させるには、0x80 を上記コードに加算してください。(MSB をドット表示に割り当てています)

それ以外のコードは、DELIMITER と NEWLINE 、ユーザプログラムモードへ移行するコード (0xaa) を除き、ブランク表示となります。

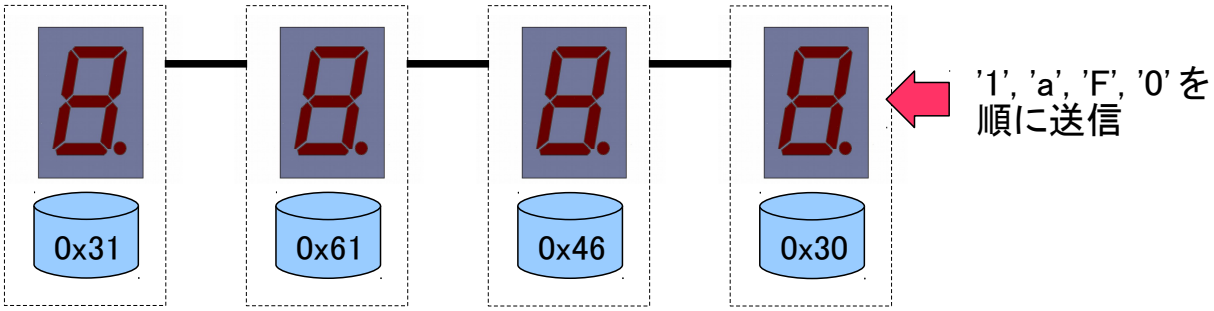
3. NEWLINE コード (0x5C: '¥') を受信した場合、表示を消灯させると同時に TX 端子から NEWLINE コードを送信します。

   さらに、バッファの内容を 0x20 に置き換えます。

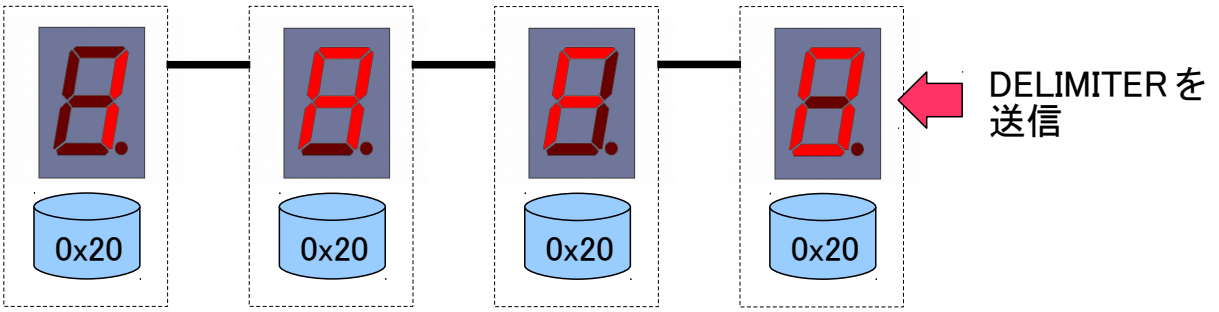


7 セグブロックを 4 個連結させたときの表示のさせ方

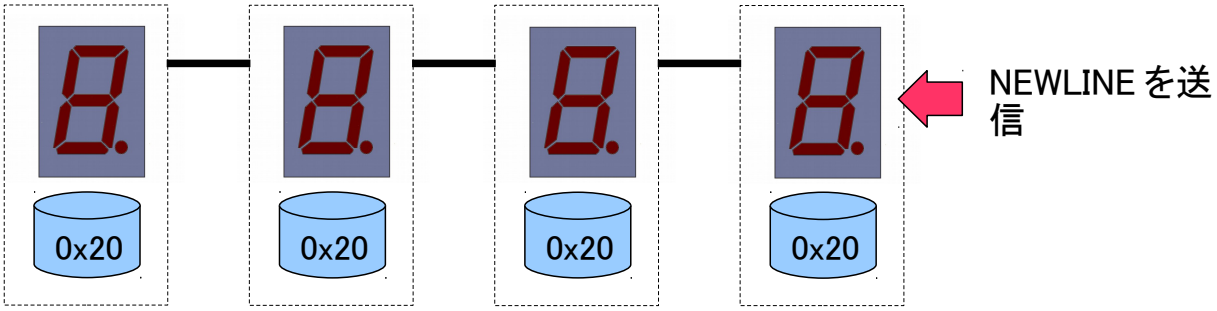
1. ASCII コードを送信
   例として '1', 'a', 'F', '0' の順に ASCII コードを送信。このとき内部バッファにデータが蓄えられるだけで 7 セグ LED の表示は変化しません。



2. DELIMITER コード送信
   連結している 7 セグ全ての表示内容が、バッファに蓄えられているデータに一斉に更新されます。上記の状態では、左から "1AF0" と 7 セグ LED が点灯します。点灯後に内部バッファが 0x20 にクリアされます。



3. NEWLINE コード送信
   連結している 7 セグ全てが一斉に消灯、内部バッファもクリアされます。



■ ユーザプログラムモード

7 セグブロックは、ユーザ自身が作成したプログラムの書き込み、実行を可能にするユーザプログラムモードを備えています。

■ ユーザプログラムの作成方法

ユーザが作成できるプログラムには下記の制約があります。

マイコン品種指定: PIC16F1823
プログラム領域: 0x001 ～ 0x5ff
リセットベクタ: 0x001
コンフィギュレーションビット:ファームウェアの設定に従う (*2)
フォーマット: Intel HEX (*3)

この制約内であれば、どの言語や環境でも作成・実行が可能です。
(*2) github のファームウェアコード参照。URL は後述。
(*3) 1 行の最長バイト数 16 、かつアドレス末尾が 0 から始まること。

MPLAB X IDE + XC8 でプログラムを作成する場合、下記の XC8-Linker の設定をすれば、通常の PIC プログラムと同様に作成できます。

Runtime -> Format hex file for download	: on
Memory model -> ROM ranges	: 1-5ff
Additional options -> Codeoffset	: 1

ユーザプログラムにコンフィギュレーションビットの記述があっても問題ありません。(書き込み時ファームウェア側で無視します)

ユーザプログラム実行時、マイコンのレジスタは初期化されず、ファームウェアで使用していた状態を引き継ぎます。必要に応じて初期化処理を行ってください。

github リポジトリ
https://github.com/oks486/bootloader_7seg_block/

ファームウェア
bootloadre_7seg_block.asm

ユーザプログラムサンプル
sample/user_program

Arduino 通信サンプル
sample/bootloader_test

■ ユーザプログラムの書き込み

ユーザプログラムを書き込むためには、ノーマルモード時に、0xaa, 0x55, 0xaa, 0xff, 0xaa を順に送信して書き込みモードに移行します。

その後、200ms 以上の待ち時間を挿入した後、intel HEX 形式のデータを送信します。1 行送信した後は、2ms 以上ウェイトを挟んでから、次の行を送信してください。

書き込みモードに移行すると、7セグ中央の 'g' セグメントのみ点灯します。書き込みが終了するとファームウェアによりリセットがかかり消灯します。

全てのプログラムデータが送信される（終了レコードを受信する）まで、ファームウェアは後続のデータを待ち続けます。

何らかのエラーが発生した場合、ドットを含めた全セグメントが点灯します。このときファームウェアが無限ループに入り、シリアルからの信号を一切受け付けなくなります。電源を切断するか、もしくは基板上のリセット端子を操作して状態を解除してください。

上記書き込みコマンドおよびプログラムデータは、接続された 7 セグブロック全てに伝搬します。そのため連結したままでの一括書き込みが可能です。

Arduino による書き込みの例は、github のコードを参照してください。

■ ユーザプログラムの実行

書き込まれたユーザプログラムを実行するためには、通常モードの時に、0xaa, 0x55, 0xaa, 0x00, 0x55 を順に送信します。その後、ユーザプログラムが実行されます。

上記実行コマンドは、接続された全てのブロックに伝搬します。

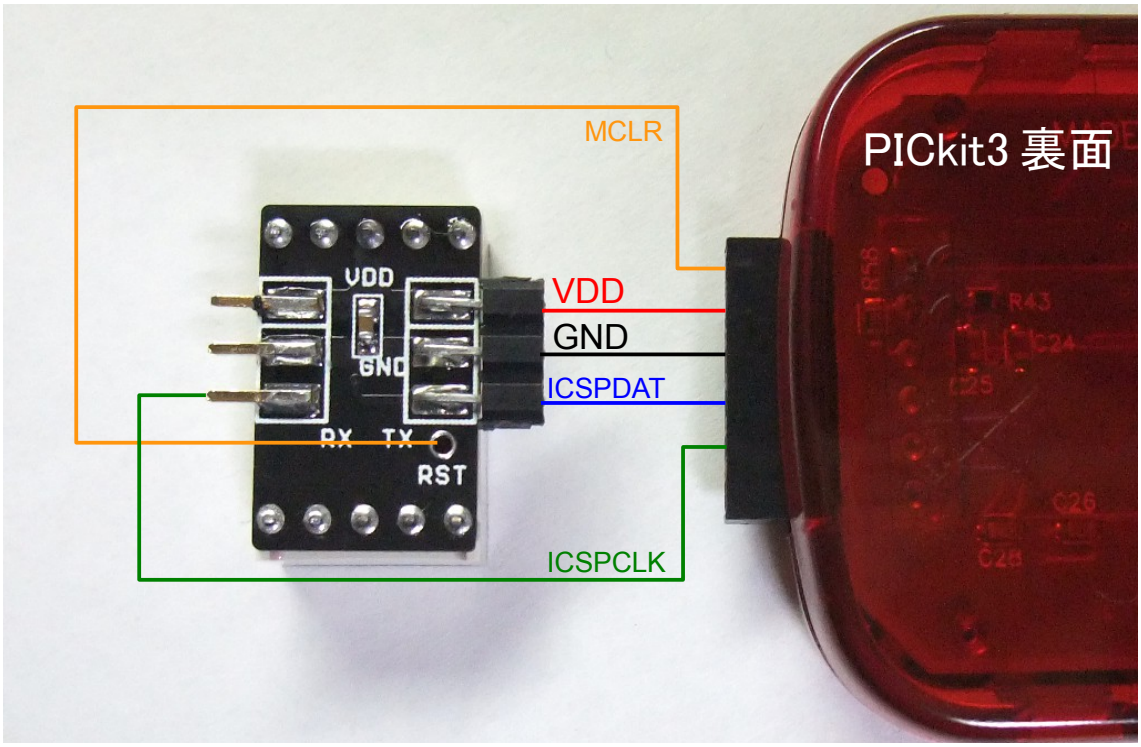
Arduino による実行の例は、github のコードを参照してください。

■ ファームウェアの書き換えについて

7 セグブロックは、既存ファームウェアのパラメータ変更や、独自に作成したファームウェアへの差し替え、また壊れたファームウェアの修復ができるようになっています。

この作業には PICkit3 等の書き込みツールと、それに対応した書き込みソフトウェアが必要になります。通常の PIC マイコンの書き換えと方法は同じです。

PICkit3 を使って 7 セグブロックを書き換える場合、接続は次のようになります。



■ 使用上の注意

- 本キットは、TSSOP パッケージおよびチップ部品の半田付けを確実にできる方、加えて基礎的な電気的知識を有する方を対象としています。
- 本キットはホビー用途として設計しています。製品に使用しないでください。
- 本キットを使用したことによって発生した、いかなる損害・損失について、当方は一切の責任を負いません。使用者本人の責任において判断してください。
- 初期不良の場合、または明らかな設計上の不具合の場合を除き、本キットの問い合わせには原則として回答できません。ご了承ください。

設計 / 著作
oaks (@oks486)

twitter : @oks486
blog : なんとかする予定 http://www.cyberchabudai.org/