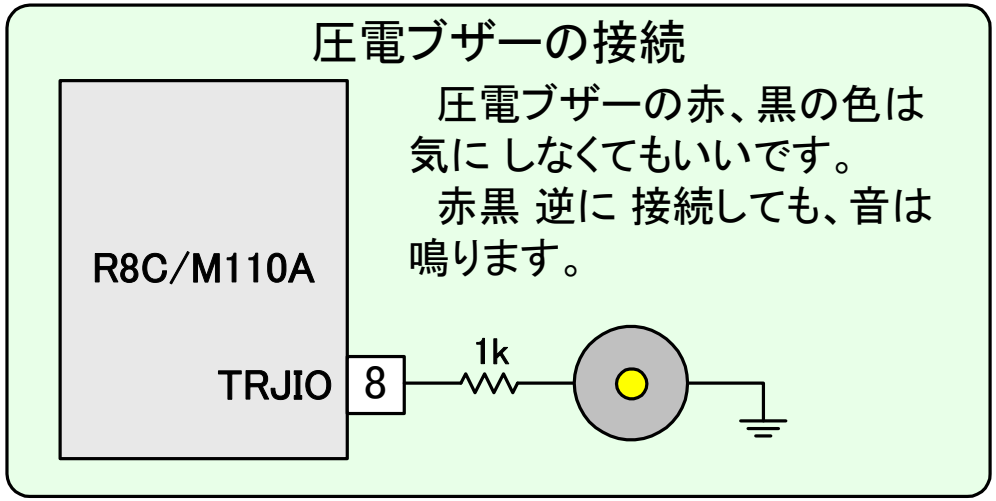
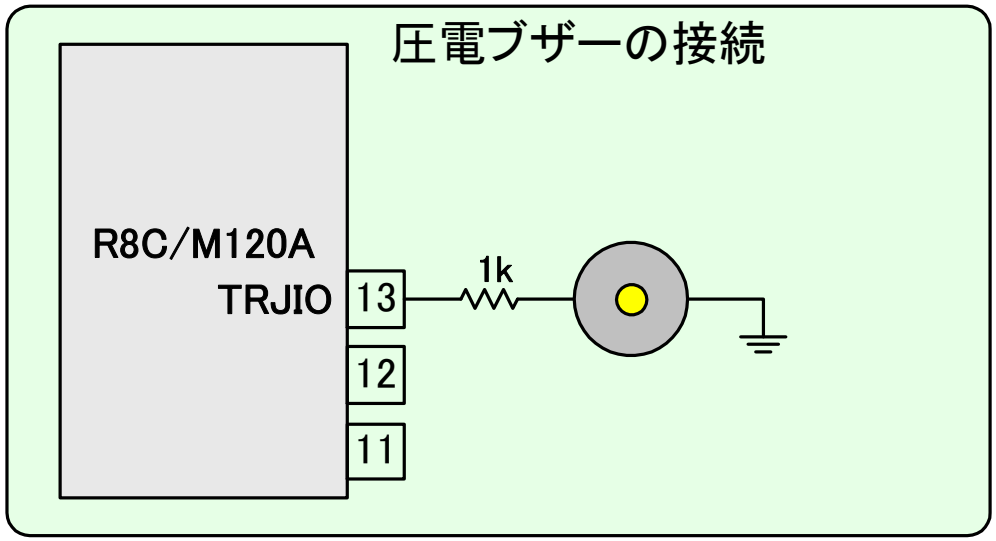
実験前に ブザー機能を追加します

以前、圧電ブザーに、ペットボトルを輪切りしたメガホンを付けて鳴らす実験をしましたが、圧電ブザーを鳴らす機能を IOCSに取り込もうと思います。 地味な機能ですが有れば便利です。

圧電ブザーを 鳴らすパルス信号は タイマー周辺回路 TRJ2を 使います。 前回は R8C/M110Aを 使用していたので R8C/M120Aは、上位互換なので、当然 M120Aでも TRJ2は 使えるという事です。

但し、TRJ2出力ポートは、M110Aと M120Aでは、足ピンの数が異なるので、ピン番号がずれてきます。TRJ2のパルス出力端子は TRJIO端子で M110Aの場合：8 ピンで、M120Aの場合：13 ピンです。 TRJIO端子に 1kΩの抵抗を入れ 圧電ブザーの どちらかのリード線を 接続します。反対側の リード線を 電源かグランドに 接続します。 右に接続に関わる 回路図を 示します。





圧電ブザー駆動の IOCS関数

圧電ブザー駆動の IOCS関数は、以下の物を考えています。

① タイマー周辺回路 TRJ2を初期化する関数を用意します。

```
void  init_trj2( void );    // TRJ2初期化
```

② 短いビープ音を 1回鳴らします。

```
void  beep1( void );    // 短いビープ音1回
```

③ 短いビープ音を 3回鳴らします。

```
void  beep3( void );    // 短いビープ音3回
```

④ 音階に準拠した音を指定長 発音する

```
void  sound_on( char *nt, WORD len, BYTE ps );
        nt  :  ノート名 ( 音の高さ )
        len :  ノート長 ( 音の長さ )
        ps  :  有効発音長 0 ～ 100 %
```

⑤ テンポ設定

```
void  set_tempo( int tp );    // テンポ設定
```

以上の関数を 用意します。ファイル名は R8CM1_IOCS_TRJ2.c とします。

以前の 101の 動画にて TRJ2のサブルーチンを見てみると、単音ですが 平均率の音程を 鳴らす機能も作ってあったので、そのまま持ってきました。

余談:

IOCSの関数を見た事がある方は、あれっ、今回は アセンブラで 作らないのですね。？と 思われるかもしれません。

今回の 音を鳴らす処理であれば、さほど処理速度の高速化を 図る必要がないと 考えたからです。確かにアセンブラは 実行速度の チューニングは 出来ますが、作るのは結構面倒ですし下手すると、バグの温床に なりかねない危険な言語とも いえます。 よって、アセンブラと、C言語は 用途によって 使い分けるのが 好ましいと思います。

実験１、I/0ポートのアクセス

最初の実験は、単純な I/Oポートアクセスの実験です。 今回は M120Aを使用するので右に M120Aの I/O ポート表を 示します。

R8Cの I/Oポートは、例えば Port1を Byte 単位でアクセスする場合は p1 と記述して アクセスします。 例えば、p1 = 0x03; と すれば、8bit の p1 ポートに 0x03を 出力した事に なります。 逆に p1ポートから byte単位で入力する場合は、sw = p1; と 記述します。

Port1の b0 に 1 を 出力する場合は p1_0 = 1; と記述します。 逆に スイッチ等が接続されていて、状態を読み出したい場合は sw1 = p1_0; と記述します。

今まで、ごく基本的な部分を説明して無かったので、初心者の方は、ここで躓いていた方もおられたかも しれませんね。 ごめんなさい。

R8C/M120AN

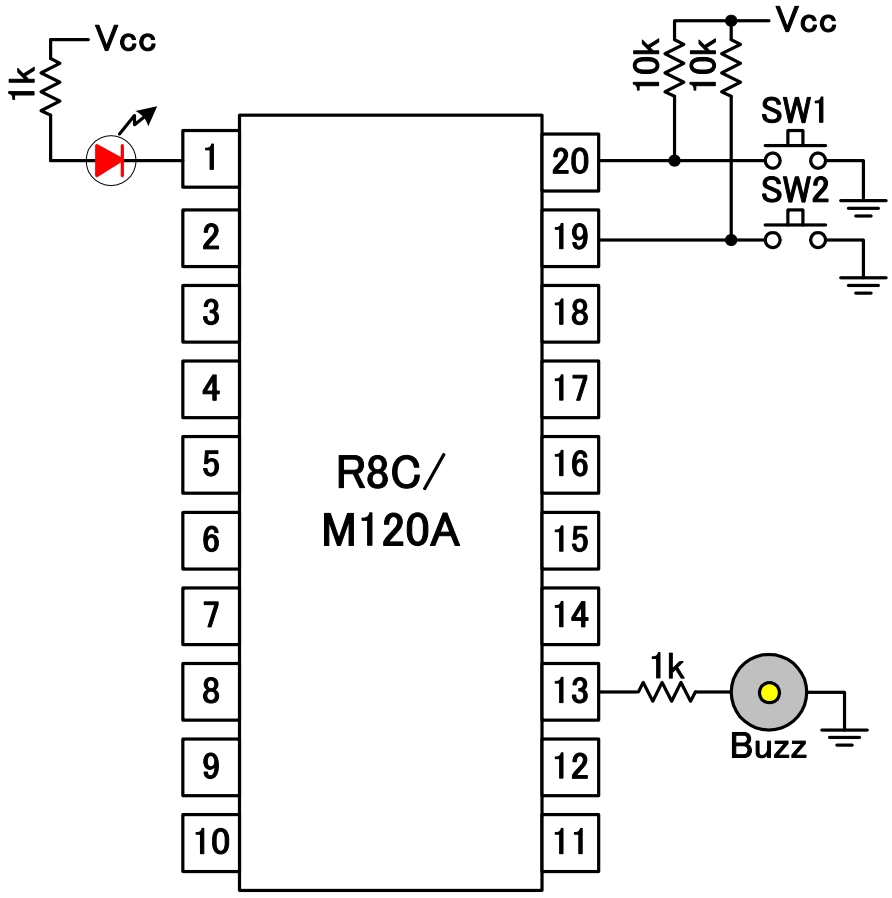
Port. 1							
b7	b6	b5	b4	b3	b2	b1	b0
P1_7	P1_6	P1_5	P1_4	P1_3	P1_2	P1_1	P1_0
13	14	15	16	17	18	19	20

Port. 3							
b7	b6	b5	b4	b3	b2	b1	b0
P3_7		P3_5	P3_4	P3_3			
2		9	10	11			

Port. 4							
b7	b6	b5	b4	b3	b2	b1	b0
P4_7	P4_6	P4_5			P4_2		
4	6	12			1		

Port. A							
b7	b6	b5	b4	b3	b2	b1	b0
							PA_0
							3

今回の実験は、I/Oポートに　押しボタンスイッチ ２個と、LED １個を 接続します。
　使用するポートは、
20ピン　p1_0 押しボタンSW1 を 接続します。
19ピン　p1_1 押しボタンSW2 を 接続します。
 1ピン　p4_2 LEDを 接続します。
SW1 と SW2 は、プルアップ抵抗 10kΩで Vcc に接続した状態にして、接点が ONすると Low に 落ちるようにします。 LEDは 1kΩの電流制限抵抗を　直列に接続し Vccに 接続して、出力ポートが　Lowに落ちた時に　点灯するようにします。 ブザーも付けておきます。
　イメージしているのは、SW1が　LEDの点灯ボタンで、SW2が、LEDの消灯ボタンという事を 想定しています。 上記のような実験を行うために、回路をどのように接続するかと、ソフトでどのように制御するかの　実験というか実習を行います。ソフトは 動画内で入力します。

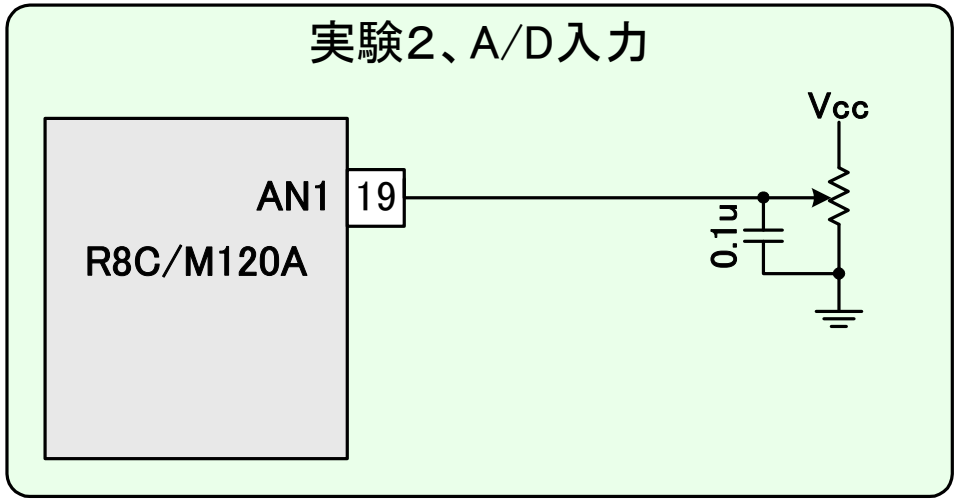


注） I/O以外の部品は書き込んでいません。

実験２、アナログデータ取り込み

２番目は、アナログ信号を マイコン内蔵の A/D変換器を使ってデジタル量子化数(数値データ)に変換します。 変換した数値データを確認のため、パソコンに 文字列データとして転送します。パソコンの受け側には TeraTerm を 使用します。サンプルレイトは 1秒に 10回で、行います。 1秒に 100回とか設定すると早すぎて、テラタームでは、滝のように文字列が流れて行きます。 逆に 1秒に 1回だと、数値の変化が ゆっくりで とらえにくい感じもします。

今回は、シンプルな例題という事で、入力は1チャネルとします。 入力端子は AN1 19ピンとします。 入力アナログ信号は 可変抵抗器 両側の端子に Vcc、Gnd を 接続して 可変抵抗器の中点を 信号源とします。
プログラムは、動画内で 入力します。



参考:

A/Dコンバータを使う上で、ミリボルト分解能のアナログ信号を取り込むので、量子化データの下位 2bit ぐらいにノイズが乗りやすいです。

対策としては、マイコンの電源を、三端子シリーズ電源を 使う アナログ信号源側にノイズフィルターを構成する。後は 例えば A/D変換を 3回連続して行い、3回のデータの合計を 3で割る平均化を行う事で、ノイズを少なく出来ます。