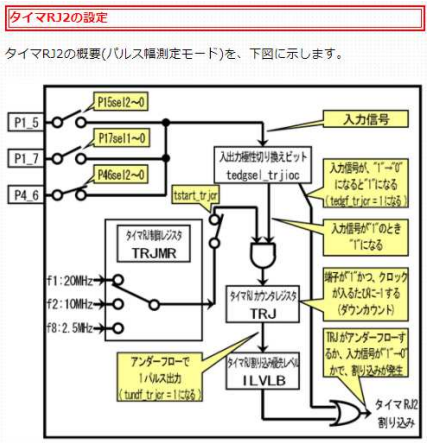
プロポ受信機から出力されるパルスを入力する　　R4.04.04

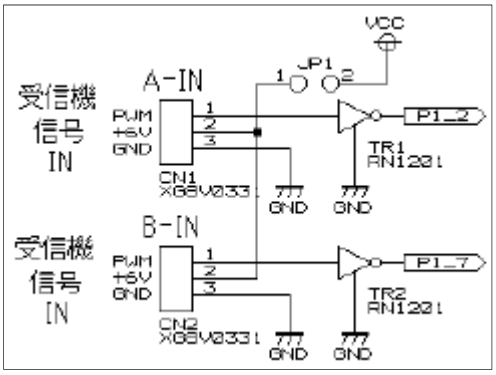
「タイマ RJ2 のパルス計測モード使えばいいんじゃない」はもっともですが、 1ch ではダメ。　最低 2ch できれば 4ch が必要です。
100 円マイコン R8C/M12A ではダメなの?と思っていたら、
ありました。マイコンカーラリー販売から販売されています。
まさに作りたかった仕様、高額なのでさっそく自作に挑戦です。



タイマRJ2を使って、入力端子(P1_5、P1_7、P4_6のどれか一つだけ選択)の"1"の時間(パルス幅)を測定します。測定時間は、f1を選んだときは3.2768msまで、f2を選んだときは6.5536ms、f8を選んだときは26.2144msになります。それ以上は、オーバーフローとなり測定できません(プログラムを工夫すればできます)。

	M-S324	プロポ信号受信リレーモータドライブ基板 プロポ受信器からの信号を受けて、リレーでモータ2個を制御することのできる基板です(最大25Aまで)。	¥5,940
	M-S325	プロポ信号受信FETモータドライブ基板 プロポ受信器からの信号を受けて、FETでモータ2個を制御することのできる基板です(最大75Aまで)。	¥11,550

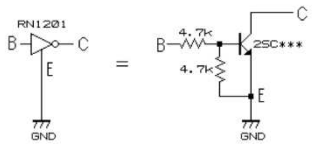
受信機の接続



※受信機の信号端子を直接接続するのではなく、
RN1201(東芝 抵抗内臓トランジスタ)を通しています。

※RN1201(TR1、TR2)について

RN1201 は抵抗を内蔵しているトランジスタです。B(ベース)に 5V を加えるとトランジスタが ON して C(コレクタ)が 0V になるため(E(エミッタ)を 0V に接続している場合)、今回の回路図では NOT 回路のように働いています。



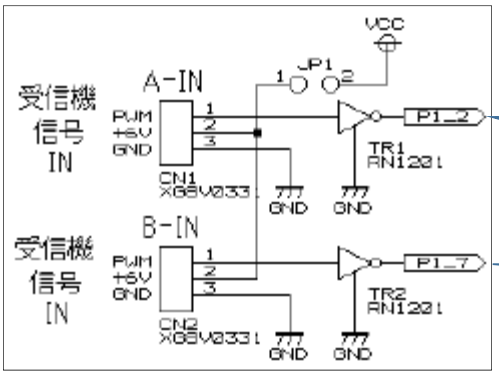
プログラムの理解

```
112 : //****************************************************************************
113 : // R8C/M12A スペシャルファンクションレジスタ(SFR)の初期化
114 : //****************************************************************************
115 : void init( void )
116 : {
117 :     // 内蔵プルアップ設定
118 :     pu1_2 = 1;                      // 受信機信号Aのトランジスタ出力
119 :     pu1_7 = 1;                      // 受信機信号Bのトランジスタ出力
120 :
121 :     // ポート出力初期設定
122 :     p1_3 = 0;                       // モータA側のFETドライバ上側
123 :     p3_4 = 0;                       // モータA側のFETドライバ下側
124 :     p3_5 = 0;                       // モータB側のFETドライバ上側
125 :     p3_7 = 0;                       // モータBA側のFETドライバ下側
126 :
127 :     // ポートの入出力設定
128 :     pd1 = 0x18;                     // ポート1の入出力設定
129 :     pd3 = 0xb0;                     // ポート3の入出力設定
130 :     pd4 = 0x00;                     // ポート4の入出力設定
131 :
132 :     // タイマRBの設定 0.5msごとに割り込みを発生させる
133 :     msttrb = 0;                     // タイマRB2を有効にする
134 :     trbmr  = 0x00;                  // 動作モード、分周比設定
135 :     trbpre = 100-1;                 // プリスケーラレジスタ
136 :     trbpr  = 100-1;                 // プライマリレジスタ
137 :     trbir  = 0xc0;                  // タイマ割込み要求
138 :     ilvlc  = 0x03;                  // 割込み優先レベル設定
139 :     trbcr  = 0x01;                  // カウント開始
140 :
```

0x18:0001 1000

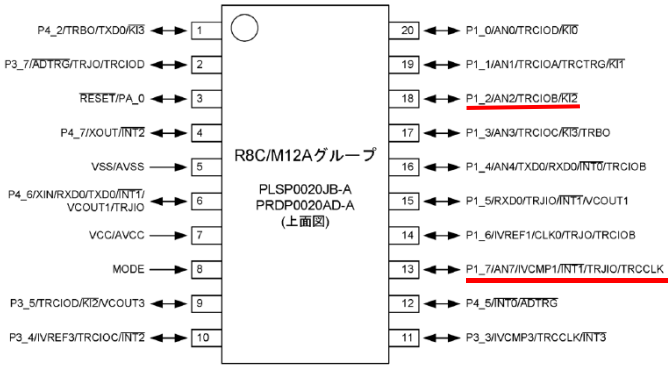
タイマ RB の割り込みは 0.5ms 間隔
割り込み処理は
intTRB()

```
141 :       // タイマRCによるPWM出力(3ch出力)
142 :       msttrc = 0;                     // タイマRCを有効にする
143 :       p12sel1 = 0;                    // TRCIOB端子の選択
144 :       p12sel0 = 1;                    // P1_2をTRCIOB端子にする
145 :       iob2_trcior0 = 1;               // TRCIOB端子はインプットキャプチャ(両エッジ)で使用
146 :       iob1_trcior0 = 1;
147 :       iob0_trcior0 = 0;
148 :       dfck1_trcdf = 1;                // デジタルフィルタ f1
149 :       dfck0_trcdf = 0;
150 :       dfb_trcdf = 1;                  // TRCIOB端子はデジタルフィルタ使用
151 :       trccr1 = 0xb0;                  // カウントソース = f8
152 :       trcgra = 25000-1;               // ｲﾝﾌﾟｯﾄｷｬﾌﾟﾁｬの周期 (1/20M*8)*25000=10ms
153 :       trcgrb = 0;                     // TRCIOB端子のON幅の設定
154 :       imiea_trcier = 1;               // trccntがオーバーフローしたときの割り込み許可
155 :       imieb_trcier = 1;               // パルスのレベルが変わったときの割り込み許可
156 :                                       // ※立ち上がり、立ち下がり、両方で割り込み発生
157 :       ilvl35 = 1;                     // 割り込みレベルの設定
158 :       ilvl34 = 0;
159 :       cts_trcmr = 1;                  // TRCCNT カウント開始
160 :
161 :       // タイマRJ2の設定
162 :       msttrj  = 0;                    // タイマRJ2を有効にする
163 :       p17sel1 = 1;                    // P1_7をTRJIO端子にする
164 :       p17sel0 = 0;                    // 〃
165 :       tedgsel_trjioc = 0;             // Lレベル幅を測定
166 :       tipf1_trjioc = 0;               // フィルタ有り f1
167 :       tipf0_trjioc = 1;
168 :       trjmr   = 0x13;                 // パルス幅測定モードに設定 f8
169 :       trj         = 0xffff;           // trjのカウンタはダウンカウント
170 :       ilvlb1 = 1;                     // 割り込みレベルの設定
171 :       ilvlb0 = 0;
172 :       trjie_trjir = 1;                // 割り込み許可
173 :       tstart_trjcr = 1;               // ﾀｲﾏｽﾀｰﾄ
174 :
175 :       // UART0の設定
176 :       mstuart = 0;                    // UART0を有効にする
177 :       p15sel2 = 0;                    // P1_5 = RXD0端子にする
178 :       p15sel1 = 0;
179 :       p15sel0 = 1;
180 :       p14sel2 = 0;                    // P1_4 = TXD0端子にする
181 :       p14sel1 = 0;
182 :       p14sel0 = 1;
183 :       u0mr = 0x05;                    // UARTﾓｰﾄﾞ8bit
184 :                                       // 1ｽﾄｯﾌﾟﾋﾞｯﾄ,ﾊﾟﾘﾃｨなし
185 :       u0c0 = 0x10;                    // U0BRGｶｳﾝﾄｿｰｽ:f1
186 :       u0brg = 129;                    // f1 / (bps*16) - 1
187 :                                       // = 20*10^6/(9600*16)-1=129.2
188 :       te_u0c1 = 1;                    // 送信許可ビット:送信許可
189 :       re_u0c1 = 1;                    // 受信許可ビット:受信許可
190 :   }
191 :
```



タイマ RC による
インプットキャプチャーモード
パルス幅計測

タイマ RJ2 による
パルス幅測定モード



タイマ RB での割り込み処理は

```
192 : //************************************************************************
193 : // タイマRB 0.5msごとの割り込み処理
194 : //************************************************************************
195 : #pragma interrupt /B intTRB(vect=24)
196 : void intTRB( void )
197 : {
198 :     static int cnt0_5ms = 0;
199 :
200 :     trbif_trbir = 0;                    // フラグのクリア
201 :
202 :     // カウンタインクリメント
203 :     cnt0_5ms++;
204 :     if( cnt_a < 30000 ) cnt_a++;
205 :     if( cnt_b < 30000 ) cnt_b++;
206 :
207 :     switch( cnt0_5ms ) {
208 :     case 1:
209 :         cnt_rb++;
210 :
211 :         // CH_A モータ制御
212 :         if( cnt_a >= 200 ) {
213 :             // 指定時間以上、パルス幅入力がなれば停止
214 :             motor_a( 0 );
215 :         } else if( pulse_a >= CH_A_PLUS_LIMIT ) {
216 :             // 上限以上なら0%
217 :             motor_a( 0 );
218 :         } else if( pulse_a <= CH_A_MINUS_LIMIT ) {
219 :             // 下限以上なら0%
220 :             motor_a( 0 );
221 :         } else if( pulse_a >= CH_A_STOP ) {
222 :             // プラスなら
223 :             motor_a( (long)(pulse_a-CH_A_STOP) * 100 / (CH_A_PLUS100-CH_A_STOP) );
224 :         } else {
225 :             // マイナスなら
226 :             motor_a( (long)(pulse_a-CH_A_STOP) * 100 / (CH_A_STOP-CH_A_MINUS100) );
227 :         }
228 :         break;
229 :
230 :     case 2:
231 :         cnt0_5ms = 0;
232 :
233 :         // CH_B モータ制御
234 :         if( cnt_b >= 200 ) {
235 :             // 指定時間以上、パルス幅入力がなれば停止
236 :             motor_b( 0 );
237 :         } else if( pulse_b >= CH_B_PLUS_LIMIT ) {
238 :             // 上限以上なら0%
239 :             motor_b( 0 );
240 :         } else if( pulse_b <= CH_B_MINUS_LIMIT ) {
241 :             // 下限以上なら0%
242 :             motor_b( 0 );
243 :         } else if( pulse_b >= CH_B_STOP ) {
244 :             // プラスなら
245 :             motor_b( (long)(pulse_b-CH_B_STOP) * 100 / (CH_B_PLUS100-CH_B_STOP) );
246 :         } else {
247 :             // マイナスなら
248 :             motor_b( (long)(pulse_b-CH_B_STOP) * 100 / (CH_B_STOP-CH_B_MINUS100) );
249 :         }
250 :         break;
251 :     }
252 : }
```

intTRB()は 0.5ms 毎実行される

0.5ms 毎にモータ A とモータ B を交互に制御

モータ A の制御

これで交互実行

モータ B の制御

タイマ RC によるパルス

```
484 :  //************************************************************************
485 :  // タイマRC 割り込み処理
486 :  //************************************************************************
487 :  #pragma interrupt intTRC(vect=7)
488 :  void intTRC( void )
489 :  {
490 :      static unsigned int pulse_a_start = 0;
491 :
492 :      if( imfb_trcsr == 1 ) {
493 :          imfb_trcsr = 0;
494 :
495 :          if( p1_2 == 0 ) {
496 :              pulse_a_start = trcgrb;
497 :          } else {
498 :              if( pulse_a_start < trcgrb ) {
499 :                  pulse_a = trcgrb - pulse_a_start;
500 :              } else {
501 :                  pulse_a = (long)25000 + trcgrb - pulse_a_start;
502 :              }
503 :              pulse_a_inc++;
504 :              cnt_a = 0;
505 :          }
506 :      }
507 :  }
508 :
```

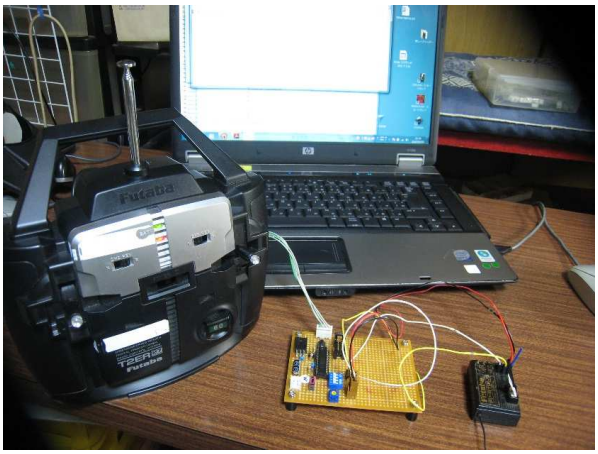
```
509 :  //************************************************************************
510 :  // タイマRJ2 割り込み処理
511 :  //************************************************************************
512 :  #pragma interrupt intTRJ2(vect=22)
513 :  void intTRJ2( void )
514 :  {
515 :      static long pulse_b_start = 0xffff;
516 :
517 :      unsigned char c;
518 :
519 :      trjif_trjir = 0;
520 :
521 :      if( tundf_trjcr == 1 ) {
522 :          // アンダーフロー
523 :          c = trjcr & 0xdf;
524 :          trjcr = c;
525 :          pulse_b_start += 0x10000;
526 :      }
527 :
528 :      if( tedgf_trjcr == 1 ) {
529 :          // 有効エッジ(立ち下がり)あり
530 :          c = trjcr & 0xef;
531 :          trjcr = c;
532 :
533 :          tstart_trjcr = 0;                   // タイマ停止
534 :          pulse_b = pulse_b_start - trj;
535 :          trj = 0xffff;
536 :          pulse_b_start = 0xffff;
537 :          tstart_trjcr = 1;                   // タイマスタート
538 :
539 :          pulse_b_inc++;
540 :          cnt_b = 0;
541 :      }
542 :  }
543 :
```

もう一度　R8C/M12A の仕様確認

パッケージ		20 ピン DIP 
動作周波数/電源電圧		～5MHz　(VCC=1.8V～5.5V) 5MHz～20MHz　(VCC=2.7V～5.5V)
メモリ	プログラム ROM	8192bytes (8KB) プログラム/イレーズ回数:10,000 回
	データフラッシュ	2048bytes (2KB) プログラム/イレーズ回数:10,000 回
	RAM	512bytes
I/O ポート		·CMOS 入出力:17 端子、プルアップ抵抗を選択可能 ·大電流ポート:8 端子
タイマ	タイマ RJ2	16 ビット×1 タイマモード、パルス出力モード(周期ごとのレベル反転出力)、イベントカウンタモード、パルス幅測定モード、パルス周期測定モード
	タイマ RB2	8 ビット×1 (8 ビットプリスケーラ付)、または 16 ビット×1 (選択可能) タイマモード、プログラマブル波形発生モード(PWM 出力)、プログラマブルワンショット発生モード、プログラマブルウェイトワンショット発生モード
	タイマ RC	16 ビット×1 (キャプチャ/コンペアレジスタ 4 本付) タイマモード(アウトプットコンペア機能、インプットキャプチャ機能)、PWM モード(出力 3 本)、PWM2 モード(PWM 出力 1 本)
A/D コンバータ		分解能:10 ビット×6 チャネル
コンパレータ B		コンパレータ B1、コンパレータ B3
通信		UART0 クロック同期形シリアル I/O／非同期形シリアル I/O 兼用
クロック発生回路		·低速オンチップ(内蔵)オシレータ:約 125kHz(リセット後のクロック) ※60～250kHz、標準は 125kHz ·高速オンチップ(内蔵)オシレータ:約 20.0MHz ※19.0～21.0MHz、標準は 20.0MHz ·XIN クロック発振回路:外付けで 2～20MHz のオシレータを搭載可能

タイマRJ2を使って、入力端子(P1_5、P1_7、P4_6のどれか一つだけ選択)の"1"の時間(パルス幅)を測定します。測定時間は、f1を選んだときは3.2768msまで、f2を選んだときは6.5536ms、f8を選んだときは26.2144msになります。それ以上は、オーバーフローとなり測定できません(プログラムを工夫すればできます)。

タイマRCのタイマモード(インプットキャプチャ機能)を使用して、TRCIOA端子に入力される外部信号のパルス幅を測定します。メイン処理で、パルス幅の測定結果を算出します。



さっそく実験

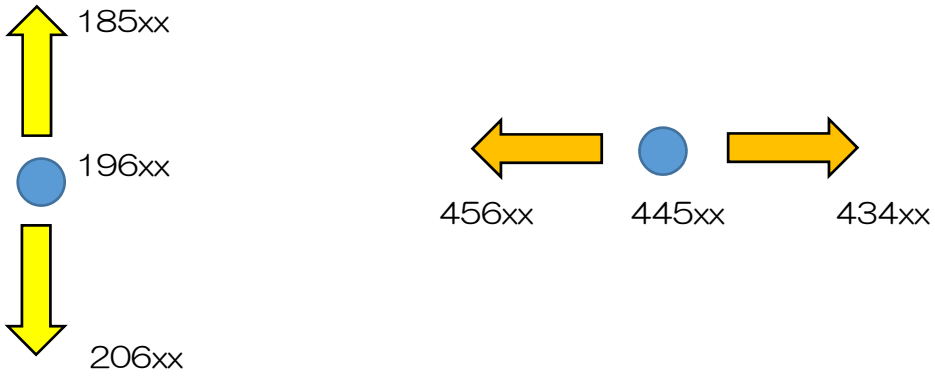
main 関数内では、TeraTerm などの通信ソフトへ、デバッグ用の内容を出力するプログラムです。
表示例を下記に示します。

```
PUROPO FET PCB Ver.1

A=03759(00297) B=03757(00297) TB6=1(1023) TB7=1(0914)
  ①    ②       ③    ④       ⑤   ⑥      ⑦   ⑧
```

①～⑧の内容について、下表に示します。

①	プロポ受信機入力コネクタ A から受信したパルスのパルス幅を表示します。
②	プロポ受信機入力コネクタ A から受信したパルスの受信回数を表示します。
③	プロポ受信機入力コネクタ B から受信したパルスのパルス幅を表示します。
④	プロポ受信機入力コネクタ B から受信したパルスの受信回数を表示します。
⑤	TB6 の入力レベルを"0"か"1"かで表示します。
⑥	TB6 に入力されている電圧を A/D 変換した値(0～1023)で表示します。 0=0V、1023=5V(電源電圧)です。アナログセンサを接続したときに、A/D 値を使います。
⑦	TB7 の入力レベルを"0"か"1"かで表示します。
⑧	TB7 に入力されている電圧を A/D 変換した値(0～1023)で表示します。 0=0V、1023=5V(電源電圧)です。アナログセンサを接続したときに、A/D 値を使います。



基礎実験はここまで。
下２桁は安定していないものの、まずはデータ取れているのであとはプログラムで何とかすれば、使えるでしょう。(動作不安定なのは、ダイレクト接続したから?)
ハードもソフトも、もう少し理解しないとダメなようです。
また 4ch にするには R8C/M12A を２セット準備すれば何とかなりそうです。