RC サーボ 24ch 基板製作キット
C 言語 RC サーボ制御プログラム
解説マニュアル

第 1.01 版
2015 年 4 月 20 日
株式会社日立ドキュメントソリューションズ

注 意 事 項 (rev.6.0H)

著作権

・本マニュアルに関する著作権は株式会社日立ドキュメントソリューションズに帰属します。
・本マニュアルは著作権法および、国際著作権条約により保護されています。

禁止事項

ユーザーは以下の内容を行うことはできません。
・第三者に対して、本マニュアルを販売、販売を目的とした宣伝、使用、営業、複製などを行うこと
・第三者に対して、本マニュアルの使用権を譲渡または再承諾すること
・本マニュアルの一部または全部を改変、除去すること
・本マニュアルを無許可で翻訳すること
・本マニュアルの内容を使用しての、人命や人体に危害を及ぼす恐れのある用途での使用

転載、複製

本マニュアルの転載、複製については、文書による株式会社日立ドキュメントソリューションズの事前の承諾が必要です。

責任の制限

本マニュアルに記載した情報は、正確を期すため、慎重に制作したものですが万一本マニュアルの記述誤りに起因する損害が生じた場合でも、株式会社日立ドキュメントソリューションズはその責任を負いません。

その他

・本マニュアルに記載の情報は本マニュアル発行時点のものであり、株式会社日立ドキュメントソリューションズは、予告なしに、本マニュアルに記載した情報または仕様を変更することがあります。製作に当たりましては、最新の内容を確認いただきますようお願いします。
・すべての商標および登録商標は、それぞれの所有者に帰属します。

連絡先

株式会社 日立ドキュメントソリューションズ
〒135-0016 東京都江東区東陽六丁目 3 番 2 号 イースト 21 タワー
E-mail：himdx.m-carrally.dd@hitachi.com

目次

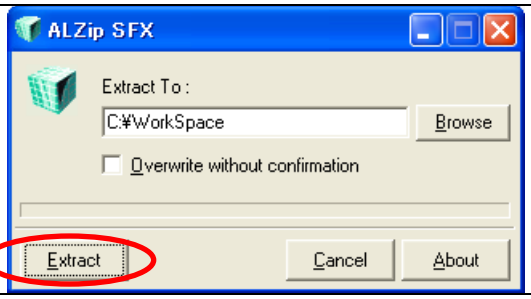
1. 概要 1
2. ワークスペースのインストール 2
3. プログラム解説「mini_mcr.c」 3
3.1 プログラムリスト 3
3.2 RC サーボの制御信号について 10
3.3 RC サーボ 24ch 基板製作キットの制御信号について 11
3.4 シンボル定義 13
3.5 メインプログラムを説明する前に 14
3.6 R8C/35A の内蔵周辺機能の初期化：init 関数 14
3.7 割り込みプログラム：intTRD0IC 関数 16
3.8 RC サーボ初期化プログラム：init_servo24ch 関数 19
3.9 RC サーボ制御プログラム：servo24ch 関数 19
3.10 メインプログラム：main 関数 20
4. 仕様 22
4.1 仕様 22
4.2 回路図 23
4.3 ポート表 24
4.4 ピン配置図 24

1. 概要

本書では、RC サーボ 24ch 基板製作キット用の C 言語 RC サーボ制御プログラムの解説を行います。RC サーボ制御プログラムは、RC サーボ 8 個とミニマイコンカーVer.2 付属のモーター2 個を制御するプログラムです。
開発環境の構築方法やプログラムのビルド、書き込みについては、「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」の「3. インストール」「4. ミニマイコンカーVer.2 の動作確認」を参照してください。

2. ワークスペースのインストール

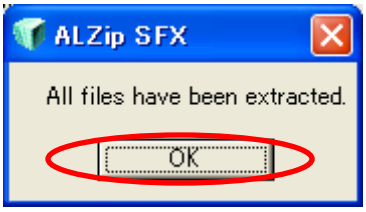
株式会社日立ドキュメントソリューションズのマイコンカーラリー販売ページからワークスペースのインストーラー「mini_mcr2_servo_24ch_vxxx.exe」（xxx はバージョン）をダウンロードします。
ダウンロードした「mini_mcr2_servo_24ch_vxxx.exe」をダブルクリックし、インストーラーを実行します。



表示されたデフォルトのインストール先のフォルダ「c:¥WorkSpace」を確認して、「Extract」をクリックします。

《**補足**》 別のフォルダを選択する場合は、「Browse」をクリックしてください。

インストールが開始されます。



ワークスペースのインストールが完了しました。「OK」をクリックします。

以上でワークスペースのインストールは完了です。

3. プログラム解説「mini_mcr.c」

WorkSpace のファイルをそのままコンパイルして書き込むと、ミニマイコンカーVer.2 が前進、後退をしながら、8 個の RC サーボが左右に動きます。

3.1 プログラムリスト

```
  1 :  //-----------------------------------------------------------------------------
  2 :  // 対象マイコン R8C/35A
  3 :  // ﾌｧｲﾙ内容     サーボ 24ch 制御
  4 :  // バージョン   Ver.1.00
  5 :  // Date         2010.06.08
  6 :  // Copyright    ルネサスマイコンカーラリー事務局
  7 :  //              日立インターメディックス株式会社
  8 :  //-----------------------------------------------------------------------------
  9 :  //-----------------------------------------------------------------------------
 10 :  // インクルード
 11 :  //-----------------------------------------------------------------------------
 12 :  #include "sfr_r835a.h"
 13 :
 14 :  //-----------------------------------------------------------------------------
 15 :  // シンボル定義
 16 :  //-----------------------------------------------------------------------------
 17 :  #define TIMER_CYCLE     155             // 1ms:0.001/(1/(20000000/128))-1
 18 :  //#define PWM_CYCLE       39999           // 16ms:0.016/(1/(20000000/8))-1
 19 :  #define PWM_CYCLE       4999           // 2ms:0.002/(1/(20000000/8))-1
 20 :
 21 :  #define Def_500Hz       4999            // 500Hz:(1/500)/(1/(20000000/8))-1
 22 :  #define Def_1000Hz      2499            // 1000Hz:(1/1000)/(1/(20000000/8))-1
 23 :
 24 :  #define Def_C3          19083           // ド:(1/131)/(1/(20000000/8))-1
 25 :  #define Def_D3          17006           // レ:(1/147)/(1/(20000000/8))-1
 26 :  #define Def_E3          15151           // ミ:(1/165)/(1/(20000000/8))-1
 27 :  #define Def_F3          14285           // ファ:(1/175)/(1/(20000000/8))-1
 28 :  #define Def_G3          12754           // ソ:(1/196)/(1/(20000000/8))-1
 29 :  #define Def_A3          11362           // ラ:(1/220)/(1/(20000000/8))-1
 30 :  #define Def_B3          10120           // シ:(1/247)/(1/(20000000/8))-1
 31 :  #define Def_C4          9541            // ド:(1/262)/(1/(20000000/8))-1
 32 :
 33 :  #define DI()            asm("FCLR I")   // 割り込み禁止
 34 :  #define EI()            asm("FSET I")   // 割り込み許可
 35 :
 36 :  #define SERVO_CENTER    1249            // 1.5ms:コンペアマッチで H なので(0.002-0.0015)/(1/(20000000/8))-1
 37 :  #define SERVO_STEP      13              // 0.5ms/90°:(0.0005/(1/(20000000/8))-1)/90
 38 :  //-----------------------------------------------------------------------------
 39 :  // 関数プロトタイプの宣言
 40 :  //-----------------------------------------------------------------------------
 41 :  void init( void );
 42 :  unsigned char sensor( void );
 43 :  void motor( int data1, int data2 );
 44 :  void timer( unsigned long timer_set );
 45 :  void beep( int data1 );
 46 :  unsigned char dipsw( void );
 47 :  unsigned char pushsw( void );
 48 :
 49 :  void init_servo24ch( void );
 50 :  void servo24ch( int ch, int angle );
 51 :
 52 :  //-----------------------------------------------------------------------------
 53 :  // グローバル変数の宣言
 54 :  //-----------------------------------------------------------------------------
 55 :  unsigned long   cnt0 = 0;               // timer 関数用
 56 :  unsigned long   cnt1 = 0;               // main 内で使用
```

```
 57 :  int             pattern = 0;            // パターン番号
 58 :
 59 :  int pwm_ch = 0;
 60 :  int pwm_enb = 0;
 61 :  unsigned short pwm_dty[24];
 62 :
 63 :  //---------------------------------------------------------------------------
 64 :  // メインプログラム
 65 :  //---------------------------------------------------------------------------
 66 :  void main(void)
 67 :  {
 68 :          int     i;
 69 :
 70 :          // 初期化
 71 :          init();
 72 :
 73 :          init_servo24ch();
 74 :
 75 :          // 起動音
 76 :          beep(Def_500Hz);
 77 :          timer(100);
 78 :          beep(Def_1000Hz);
 79 :          timer(100);
 80 :          beep(0);
 81 :
 82 :          while( pushsw() == 0 ){
 83 :          }
 84 :
 85 :          while(1){
 86 :
 87 :  //              servo24ch( 0, 90 );
 88 :  //              servo24ch( 1, 90 );
 89 :  //              servo24ch( 2, 90 );
 90 :  //              servo24ch( 3, 90 );
 91 :  //              servo24ch( 4, 90 );
 92 :  //              servo24ch( 5, 90 );
 93 :  //              servo24ch( 6, 90 );
 94 :  //              servo24ch( 7, 90 );
 95 :
 96 :  //              servo24ch( 8, 90 );
 97 :  //              servo24ch( 9, 90 );
 98 :  //              servo24ch( 10, 90 );
 99 :  //              servo24ch( 11, 90 );
100 :  //              servo24ch( 12, 90 );
101 :  //              servo24ch( 13, 90 );
102 :  //              servo24ch( 14, 90 );
103 :  //              servo24ch( 15, 90 );
104 :
105 :                  servo24ch( 16, 90 );
106 :                  servo24ch( 17, 90 );
107 :                  servo24ch( 18, 90 );
108 :                  servo24ch( 19, 90 );
109 :                  servo24ch( 20, 90 );
110 :                  servo24ch( 21, 90 );
111 :                  servo24ch( 22, 90 );
112 :                  servo24ch( 23, 90 );
113 :
114 :                  motor( 100, 100 );
115 :
116 :                  timer(1000);
117 :
118 :  //              servo24ch( 0, -90 );
119 :  //              servo24ch( 1, -90 );
120 :  //              servo24ch( 2, -90 );
121 :  //              servo24ch( 3, -90 );
122 :  //              servo24ch( 4, -90 );
123 :  //              servo24ch( 5, -90 );
124 :  //              servo24ch( 6, -90 );
125 :  //              servo24ch( 7, -90 );
126 :
127 :  //              servo24ch( 8, -90 );
128 :  //              servo24ch( 9, -90 );
```

```
129 :  //              servo24ch( 10, -90 );
130 :  //              servo24ch( 11, -90 );
131 :  //              servo24ch( 12, -90 );
132 :  //              servo24ch( 13, -90 );
133 :  //              servo24ch( 14, -90 );
134 :  //              servo24ch( 15, -90 );
135 :
136 :                  servo24ch( 16, -90 );
137 :                  servo24ch( 17, -90 );
138 :                  servo24ch( 18, -90 );
139 :                  servo24ch( 19, -90 );
140 :                  servo24ch( 20, -90 );
141 :                  servo24ch( 21, -90 );
142 :                  servo24ch( 22, -90 );
143 :                  servo24ch( 23, -90 );
144 :
145 :                  motor( -100, -100 );
146 :
147 :                  timer(1000);
148 :
149 :          }
150 :
151 :  }
152 :
153 :  //------------------------------------------------------------------------
154 :  // R8C/35A の内蔵周辺機能の初期化
155 :  //------------------------------------------------------------------------
156 :  void init( void )
157 :  {
158 :          unsigned char i = 0;
159 :
160 :          // 割り込み禁止
161 :          DI();
162 :
163 :          // クロック発生回路の XIN クロック設定
164 :          prc0 = 1;
165 :
166 :          cm13 = 1;
167 :          cm05 = 0;
168 :          while(i <= 50) i++;
169 :          ocd2 = 0;
170 :
171 :          prc0 = 0;
172 :
173 :          // I/O ポートの入出力設定
174 :          prc2 = 1;                       // pd0 レジスタへの書き込み許可
175 :          pd0 = 0xe0;                     // P0_0～P0_3:センサー
176 :                                          // P0_4:マイクロスイッチ
177 :                                          // P0_5～P0_7:LED
178 :          prc2 = 0;                       // pd0 レジスタへの書き込み禁止
179 :          pd1 = 0xdf;                     // P1_0～P1_3:LED
180 :                                          // P1_4:TXD0
181 :                                          // P1_5:RXD0
182 :          pd2 = 0xfe;                     // P2_0:スイッチ
183 :                                          // P2_1:AIN1
184 :                                          // P2_2:PWMA
185 :                                          // P2_3:BIN1
186 :                                          // P2_4:PWMB
187 :                                          // P2_5:SERVO
188 :                                          // P2_6:AIN2
189 :                                          // P2_7:BIN2
190 :          pd3 = 0xfb;                     // P3_2:赤外線受信
191 :                                          // P3_4:ブザー
192 :          pd4 = 0x80;                     // P4_2:VREF
193 :                                          // P4_3～P4_5:DIPSW
194 :                                          // P4_6:XIN
195 :                                          // P4_7:XOUT
196 :          pd5 = 0x40;                     // P5_7:DIPSW
197 :          pd6 = 0xff;                     //
198 :
199 :
200 :
```

```
201 :          mstcr = 0x00;                    // モジュールストップ解除
202 :
203 :
204 :
205 :          // タイマ RB の 1ms 割り込み設定
206 :          trbmr = 0x00;                    // カウントソースは f1
207 :          trbpre = 128 - 1;                // プリスケーラ
208 :          trbpr = TIMER_CYCLE;             // プライマリカウンタ
209 :          trbic = 0x01;                    // タイマ RB の割り込みレベル設定
210 :          trbcr = 0x01;                    // カウントを開始
211 :
212 :          // タイマ RC の PWM モード
213 :          trccr1 = 0xb0;                   // カウントソースは f8
214 :          trcgra = 0;                      // 圧電サウンダの周期
215 :          trcgrc = 0;                      // 圧電サウンダのデューティ比
216 :          trccr2 = 0x02;                   // TRCIOC 端子はアクティブレベル H
217 :          trcoer = 0x0b;                   // TRCIOC 端子の出力許可
218 :          trcpsr1 = 0x02;                  // TRCIOC 端子を P3_4 に割り当て
219 :          trcmr = 0x8a;                    // カウントを開始
220 :
221 :          // タイマ RD のリセット同期 PWM モード
222 :          trdpsr0 = 0x08;                  // TRDIOB0 端子を P2_2 に割り当て
223 :          trdpsr1 = 0x05;                  // TRDIOB1 端子を P2_5 に割り当て
224 :                                           // TRDIOA1 端子を P2_4 に割り当て
225 :          trdmr = 0xf0;                    // レジスタをバッファ動作にする
226 :  //        trdfcr = 0x01;                   // リセット同期 PWM モードに設定
227 :          trdfcr = 0x0d;                   // リセット同期 PWM モードに設定
228 :          trdoer1 = 0xcd;                  // TRDIOB1 の出力許可
229 :                                           // TRDIOA1 の出力許可
230 :                                           // TRDIOB0 端子の出力許可
231 :          trdcr0 = 0x23;                   // カウントソースは f8
232 :          trdgra0 = trdgrc0 = PWM_CYCLE;   // 周期
233 :  //        trdgrb0 = trdgrd0 = 0;           // TRDIOB0 端子（左モータ）
234 :  //        trdgra1 = trdgrc1 = 0;           // TRDIOA1 端子（右モータ）
235 :  //        trdgrb1 = trdgrd1 = 0;           // TRDIOB1 端子（サーボ）
236 :          trdgrb0 = trdgrd0 = PWM_CYCLE+1;          // TRDIOB0 端子（左モータ）
237 :          trdgra1 = trdgrc1 = PWM_CYCLE+1;          // TRDIOA1 端子（右モータ）
238 :          trdgrb1 = trdgrd1 = PWM_CYCLE+1;          // TRDIOB1 端子（サーボ）
239 :
240 :          trdier0 = 0x01;// 割り込み許可
241 :          trd0ic = 0x01;// 割り込みレベル設定
242 :
243 :          trdstr = 0x0d;                   // カウントを開始
244 :
245 :          // 割り込み許可
246 :          EI();
247 :  }
248 :
249 :  //------------------------------------------------------------------------
250 :  // 割り込み
251 :  //------------------------------------------------------------------------
252 :  #pragma interrupt intTRBIC (vect=24)
253 :  void intTRBIC( void )
254 :  {
255 :          p0_7 = ~p0_7;
256 :
257 :          if( p0_7 == 0 ){
258 :                  //p0_1、p0_3 のモニタが可能
259 :                  p0_5 = ~p0_1;
260 :                  p0_6 = ~p0_3;
261 :          }else{
262 :                  //p0_0、p0_2 のモニタが可能
263 :                  p0_5 = p0_0;
264 :                  p0_6 = p0_2;
265 :          }
266 :
267 :          cnt0++;
268 :          cnt1++;
269 :  }
270 :
271 :  //------------------------------------------------------------------------
272 :  // 割り込み
```

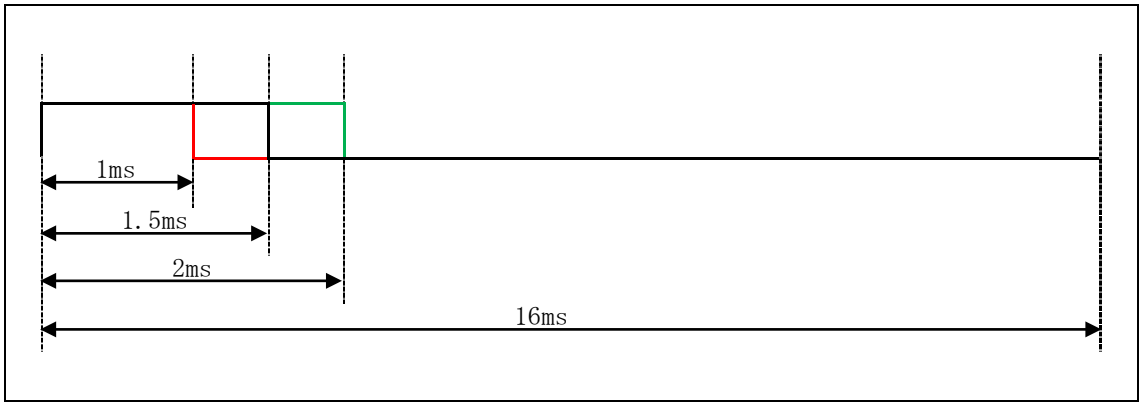
```
273 :  //------------------------------------------------------------------------
274 :  #pragma interrupt intTRD0IC (vect=8)
275 :  void intTRD0IC( void )
276 :  {
277 :          imfa_trdsr0 = 0;
278 :
279 :          if( pwm_enb != 0 ){
280 :  //              trdgrd0 = pwm_dty[pwm_ch];      // モータードライバ部分を切り離した場合はコメント解除
281 :  //              trdgrc1 = pwm_dty[8+pwm_ch];    // モータードライバ部分を切り離した場合はコメント解除
282 :                  trdgrd1 = pwm_dty[16+pwm_ch];
283 :
284 :
285 :                  switch( pwm_ch ){
286 :                  case 1:
287 :                          p6 = 0x01;
288 :                          break;
289 :                  case 2:
290 :                          p6 = 0x02;
291 :                          break;
292 :                  case 3:
293 :                          p6 = 0x04;
294 :                          break;
295 :                  case 4:
296 :                          p6 = 0x08;
297 :                          break;
298 :                  case 5:
299 :                          p6 = 0x10;
300 :                          break;
301 :                  case 6:
302 :                          p6 = 0x20;
303 :                          break;
304 :                  case 7:
305 :                          p6 = 0x40;
306 :                          break;
307 :                  case 0:
308 :                          p6 = 0x80;
309 :                          break;
310 :                  }
311 :
312 :                  pwm_ch++;
313 :                  if( 7 < pwm_ch ){
314 :                          pwm_ch = 0;
315 :                  }
316 :          }
317 :  }
318 :
319 :  //------------------------------------------------------------------------
320 :  // センサー状態検出
321 :  // 引数         なし
322 :  // 戻り値       センサ値
323 :  //------------------------------------------------------------------------
324 :  unsigned char sensor( void )
325 :  {
326 :          volatile unsigned char  data1;
327 :
328 :          data1 = ~p0;                    // ラインの色は白
329 :          data1 = data1 & 0x0f;
330 :
331 :          return( data1 );
332 :  }
333 :
334 :  //------------------------------------------------------------------------
335 :  // モーター速度制御
336 :  // 引数         左モータ:-100～100、右モータ:-100～100
337 :  //              0 で停止、100 で正転 100%、-100 で逆転 100%
338 :  // 戻り値       なし
339 :  //------------------------------------------------------------------------
340 :  void motor( int data1, int data2 )
341 :  {
342 :          volatile int    motor_r;
343 :          volatile int    motor_l;
344 :          volatile int    sw_data;
```

```
345 :
346 :          sw_data = dipsw() + 5;
347 :          motor_l = (long)data1 * sw_data / 20;
348 :          motor_r = (long)data2 * sw_data / 20;
349 :
350 :          if( motor_l >= 0 ) {
351 :                  p2_1 = 0;
352 :                  p2_6 = 1;
353 :  //                trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
354 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) - (long)( PWM_CYCLE - 1 ) * motor_l / 100;
355 :          } else {
356 :                  p2_1 = 1;
357 :                  p2_6 = 0;
358 :  //                trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
359 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) - (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
360 :          }
361 :
362 :          if( motor_r >= 0 ) {
363 :                  p2_3 = 0;
364 :                  p2_7 = 1;
365 :  //                trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
366 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) - (long)( PWM_CYCLE - 1 ) * motor_r / 100;
367 :          } else {
368 :                  p2_3 = 1;
369 :                  p2_7 = 0;
370 :  //                trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
371 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) - (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
372 :          }
373 :  }
374 :
375 :  //---------------------------------------------------------------------------
376 :  // 時間稼ぎ
377 :  // 引数          タイマ値 1=1ms
378 :  // 戻り値        なし
379 :  //---------------------------------------------------------------------------
380 :  void timer( unsigned long data1 )
381 :  {
382 :          cnt0 = 0;
383 :          while( cnt0 < data1 );
384 :  }
385 :
386 :  //---------------------------------------------------------------------------
387 :  // 音を鳴らす
388 :  // 引数          (1/音の周波数)/(1/(クロック周波数/8))-1
389 :  // 戻り値        なし
390 :  //---------------------------------------------------------------------------
391 :  void beep( int data1 )
392 :  {
393 :          trcgra = data1;                 // 周期の設定
394 :          trcgrc = data1 / 2;             // デューティ 50%のため周期の半分の値
395 :  }
396 :
397 :  //---------------------------------------------------------------------------
398 :  // DIP スイッチ状態検出
399 :  // 引数          なし
400 :  // 戻り値        0～15、DIP スイッチが ON の場合、対応するビットが 0 になります。
401 :  //---------------------------------------------------------------------------
402 :  unsigned char dipsw( void )
403 :  {
404 :          volatile unsigned char  data1;
405 :
406 :          data1 = ( ( p5 >> 4 ) & 0x08 ) | ( ( p4 >> 3 ) & 0x07 );
407 :
408 :          return( data1 );
409 :  }
410 :
411 :  //---------------------------------------------------------------------------
412 :  // プッシュスイッチ状態検出
413 :  // 引数          なし
414 :  // 戻り値        スイッチが押されていない場合:0、押された場合:1
415 :  //---------------------------------------------------------------------------
416 :  unsigned char pushsw( void )
```

```
417 :  {
418 :          unsigned char data1;
419 :
420 :          data1 = ~p2;
421 :          data1 &= 0x01;
422 :
423 :          return( data1 );
424 :  }
425 :
426 :  //---------------------------------------------------------------------------
427 :  // サーボ初期化
428 :  // 引数         なし
429 :  // 戻り値       なし
430 :  //---------------------------------------------------------------------------
431 :  void init_servo24ch( void )
432 :  {
433 :          int i;
434 :
435 :          for( i = 0; i < 24; i++ ){
436 :                  pwm_dty[i] = PWM_CYCLE+1;
437 :          }
438 :
439 :          pwm_enb = 1;
440 :
441 :  }
442 :
443 :  //---------------------------------------------------------------------------
444 :  // サーボ制御関数
445 :  // 引数         チャンネル:0～23、角度
446 :  // 戻り値       なし
447 :  //---------------------------------------------------------------------------
448 :  void servo24ch( int ch, int angle )
449 :  {
450 :          int i;
451 :
452 :          if( 0 <= ch && ch < 24 ){
453 :                  i = SERVO_CENTER + SERVO_STEP * angle;
454 :                  if( 0 <= i && i < PWM_CYCLE ){
455 :                          pwm_dty[ch] = i;
456 :                  }
457 :          }
458 :
459 :  }
```

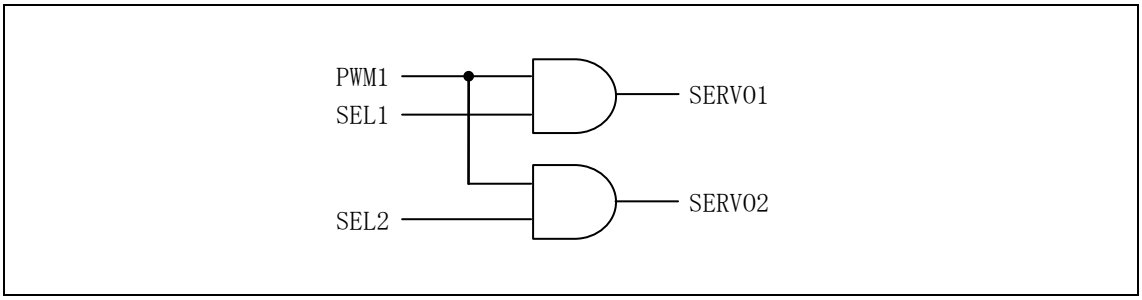
3.2 RC サーボの制御信号について

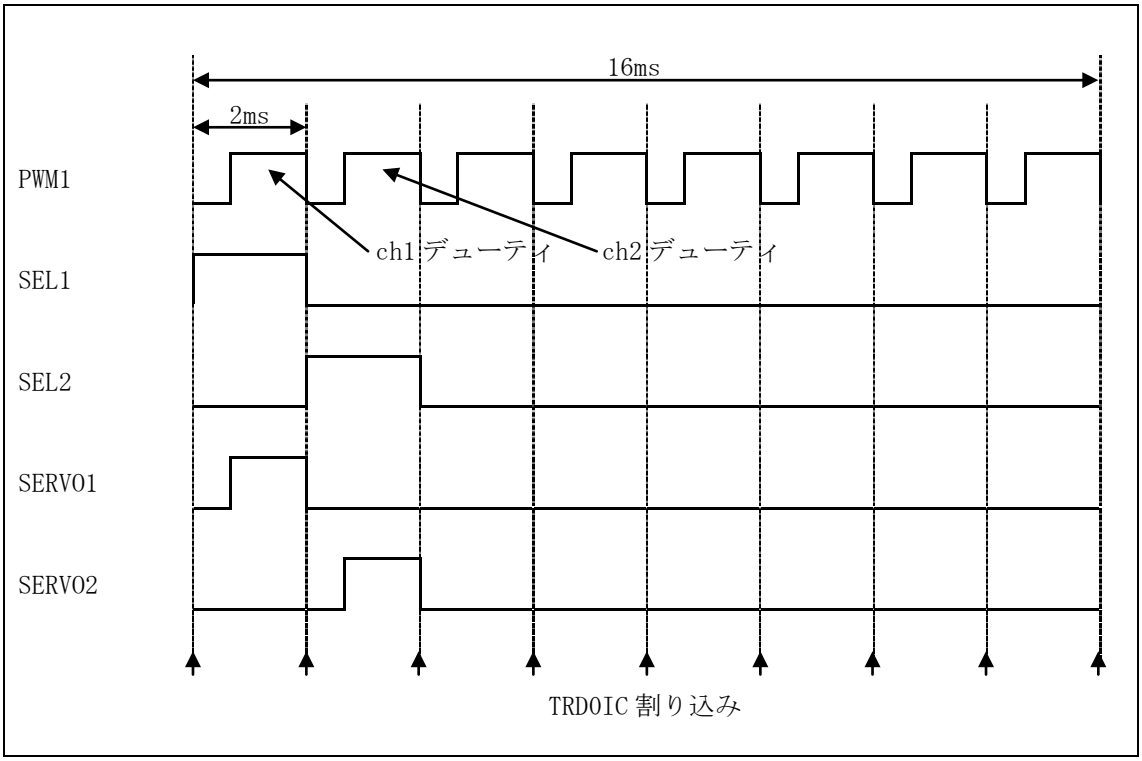
RC サーボは 16ms～20ms の周期で、H レベル幅を 1.5ms を中心として±0.5ms 変化させると中心から±90° 回転するようになっています（メーカーによって異なります)。



3.3 RC サーボ 24ch 基板製作キットの制御信号について

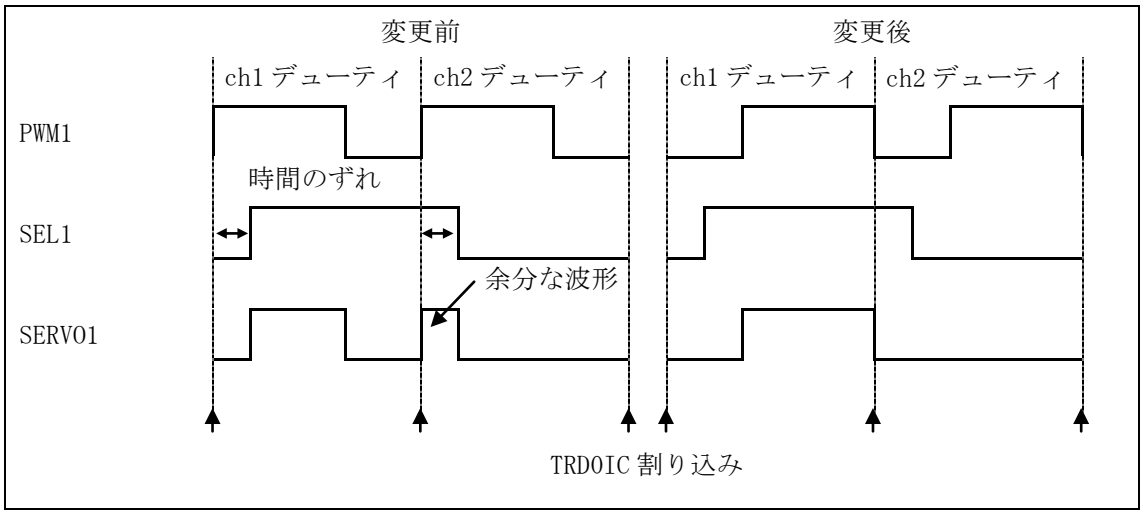
ミニマイコンカーVer.2 のポート 2 から出ている 3 本の PWM 信号（モーター×2、RC サーボ×1）をそれぞれ 8 本に分配して 24ch を制御しています。以下に 2ch 分の分配の回路と波形を示します。





H レベル幅の最大の 2ms ごとの割り込みで、デューティと選択信号の切り替えを行っています。8ch に分配するので、RC サーボの周期は 16ms になります。

「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」で使用したプログラムでは PWM を初期出力 H、アクティブレベル L にしていますが、このプログラムでは、初期出力 L、アクティブレベル H にしています。
これは、タイマーRD がコンペアマッチで PWM を初期出力に戻すのと、割り込みで選択信号の切り替えをソフトで行うのとに時間のずれがあり、このずれのため次のチャネルの PWM が前のチャネルにも一瞬出てしまうのを防ぐためです。以下に変更前と変更後の波形を示します。
※変更後の場合でも、デューティが 100%のときには余分な波形が出てしまうので注意してください。



3.4　シンボル定義

プログラム

```
 18 :  //#define PWM_CYCLE       39999          // 16ms:0.016/(1/(20000000/8))-1
 19 :  #define PWM_CYCLE       4999          // 2ms:0.002/(1/(20000000/8))-1

 36 :  #define SERVO_CENTER    1249           // 1.5ms:コンペアマッチでHなので(0.002-0.0015)/(1/(20000000/8))-1
 37 :  #define SERVO_STEP      13             // 0.5ms/90°:(0.0005/(1/(20000000/8))-1)/90
```

名称	説明
PWM_CYCLE	PWM_CYCLE は、左右モーターに加えるタイマーRD の PWM 周期を設定します。 今回は 2［ms］に設定しますので、 （2×10^{-3}）÷（1÷（$20\times10^{6}\div8$））－1＝4999 となります。
SERVO_CENTER	SERVO_CENTER は RC サーボがセンターを向く H レベル幅を設定します。 今回は 1.5［ms］に設定しますので、 （2－1.5）×10^{-3}÷（1÷（$20\times10^{6}\div8$））－1＝1249 となります。
SERVO_STEP	SERVO_STEP は RC サーボの 1° に相当する H レベル幅を設定します。 今回は 0.5÷90［ms］に設定しますので、 0.5×10^{-3}÷90÷（1÷（$20\times10^{6}\div8$））－1＝13 となります。 ※SERVO_STEP×角度が SERVO_CENTER より小さい値にならないようにしてください。

3.5　メインプログラムを説明する前に

main 関数は、main 関数の後に記されている関数を組み合わせてプログラムしていますので、先に main 関数以外の関数の解説を行います。

RC サーボ制御以外の関数については「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」の「5. プログラム解説「mini_mcr.c」」を参照してください。

3.6　R8C/35A の内蔵周辺機能の初期化：init 関数

init 関数は「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」で解説されています。ここでは、init 関数の変更部分のみを解説します。

```
226 :  //        trdfcr = 0x01;                  // リセット同期 PWM モードに設定
227 :           trdfcr = 0x0d;                 // リセット同期 PWM モードに設定

233 :  //        trdgrb0 = trdgrd0 = 0;          // TRDIOB0 端子（左モータ）
234 :  //        trdgra1 = trdgrc1 = 0;          // TRDIOA1 端子（右モータ）
235 :  //        trdgrb1 = trdgrd1 = 0;          // TRDIOB1 端子（サーボ）
236 :           trdgrb0 = trdgrd0 = PWM_CYCLE+1;          // TRDIOB0 端子（左モータ）
237 :           trdgra1 = trdgrc1 = PWM_CYCLE+1;          // TRDIOA1 端子（右モータ）
238 :           trdgrb1 = trdgrd1 = PWM_CYCLE+1;          // TRDIOB1 端子（サーボ）

240 :           trdier0 = 0x01;// 割り込み許可
241 :           trd0ic = 0x01;// 割り込みレベル設定
```

タイマーRD を、初期出力 L コンペアマッチで H 出力、周期ごとに割り込みに設定しています。

レジスタ	ビット	シンボル	説明	設定値
TRDFCR	7	PWM3	リセット同期 PWM モードでは無効なので、“0”にします。	0x0d
	6	STCLK	外部クロック入力を無効にするので、“0”にします。	
	5	ADEG	リセット同期 PWM モードでは無効なので、“0”にします。	
	4	ADTRG		
	3	OLS1	初期出力 L、アクティブレベル H にしますので、“11”にします。	
	2	OLS0		
	1	CMD1	リセット同期 PWM モードでは、“01”にします。	
	0	CMD0		
TRDGRB0 TRDGRD0	15-0		PWM 端子から L を出力するため、“PWM_CYCLE+1”にします。	PWM_CYCLE+1
TRDGRA1 TRDGRC1	15-0		PWM 端子から L を出力するため、“PWM_CYCLE+1”にします。	PWM_CYCLE+1
TRDGRB1 TRDGRD1	15-0		PWM 端子から L を出力するため、“PWM_CYCLE+1”にします。	PWM_CYCLE+1
TRDIER0	7	-	何も配置されていないので、“0”にします。	0x01
	6	-		
	5	-		
	4	OVIE	OVF ビットによる割り込みを禁止にするため、“0”にします。	
	3	IMIED	IMFD ビットによる割り込みを禁止にするため、“0”にします。	
	2	IMIEC	IMFC ビットによる割り込みを禁止にするため、“0”にします。	
	1	IMIEB	IMFB ビットによる割り込みを禁止にするため、“0”にします。	
	0	IMIEA	IMFA ビットによる割り込みを許可にするため、“1”にします。	
TRD0IC	7	-	何も配置されていないので、“0”にします。	0x01
	6	-		
	5	-		
	4	-		
	3	IR	読み出し専用なので“0”にします。	
	2	ILVL2	割り込みレベルを 1 にするため、“001”にします。	
	1	ILVL1		
	0	ILVL0		

3.7 割り込みプログラム：intTRD0IC 関数

intTRD0IC 関数は周期コンペアマッチ時に割り込みで実行されます。

プログラム

```
274 :  #pragma interrupt intTRD0IC (vect=8)
275 :  void intTRD0IC( void )
276 :  {
277 :          imfa_trdsr0 = 0;
278 :
279 :          if( pwm_enb != 0 ){
280 :  //              trdgrd0 = pwm_dty[pwm_ch];      // モータードライバ部分を切り離した場合はコメント解除
281 :  //              trdgrc1 = pwm_dty[8+pwm_ch];    // モータードライバ部分を切り離した場合はコメント解除
282 :                  trdgrd1 = pwm_dty[16+pwm_ch];
283 :
284 :
285 :                  switch( pwm_ch ){
286 :                  case 1:
287 :                          p6 = 0x01;
288 :                          break;
289 :                  case 2:
290 :                          p6 = 0x02;
291 :                          break;
292 :                  case 3:
293 :                          p6 = 0x04;
294 :                          break;
295 :                  case 4:
296 :                          p6 = 0x08;
297 :                          break;
298 :                  case 5:
299 :                          p6 = 0x10;
300 :                          break;
301 :                  case 6:
302 :                          p6 = 0x20;
303 :                          break;
304 :                  case 7:
305 :                          p6 = 0x40;
306 :                          break;
307 :                  case 0:
308 :                          p6 = 0x80;
309 :                          break;
310 :                  }
311 :
312 :                  pwm_ch++;
313 :                  if( 7 < pwm_ch ){
314 :                          pwm_ch = 0;
315 :                  }
316 :          }
317 :  }
```

```
279 :          if( pwm_enb != 0 ){
```

pwm_enb 変数が 0 の場合は、制御を行いません。

```
280 :  //              trdgrd0 = pwm_dty[pwm_ch];      // モータードライバ部分を切り離した場合はコメント解除
281 :  //              trdgrc1 = pwm_dty[8+pwm_ch];    // モータードライバ部分を切り離した場合はコメント解除
282 :                  trdgrd1 = pwm_dty[16+pwm_ch];
```

pwm_dty 変数の値を trdgrd0、trdgrc1、trdgrd1 レジスタにセットして、TRDIOB0、TRDIOA1、TRDIOB1 端子から任意のデューティの波形を出力します。モーターを切り離さない場合は E、F グループの 8ch しか使えませんので、A、B、C、D グループへ出力する部分はコメントにしています。

```
285 :                   switch( pwm_ch ){
286 :                   case 1:
287 :                           p6 = 0x01;
288 :                           break;
289 :                   case 2:
290 :                           p6 = 0x02;
291 :                           break;
292 :                   case 3:
293 :                           p6 = 0x04;
294 :                           break;
295 :                   case 4:
296 :                           p6 = 0x08;
297 :                           break;
298 :                   case 5:
299 :                           p6 = 0x10;
300 :                           break;
301 :                   case 6:
302 :                           p6 = 0x20;
303 :                           break;
304 :                   case 7:
305 :                           p6 = 0x40;
306 :                           break;
307 :                   case 0:
308 :                           p6 = 0x80;
309 :                           break;
310 :                   }
```

pwm_ch 変数の値によってポート 6 の選択信号を切り替えます。数値と対応するビットがひとつずれていますが、これは、デューティのレジスタをバッファ動作にしているためです。

```
312 :                   pwm_ch++;
313 :                   if( 7 < pwm_ch ){
314 :                           pwm_ch = 0;
315 :                   }
```

pwm_ch 変数をカウントアップして 0～7 の値を繰り返す処理をしています。

pwm_dty 変数の配列番号とコネクタの対応

配列番号	コネクタ	
0	A	CN24
1		CN22
2		CN20
3		CN18
4	B	CN25
5		CN23
6		CN21
7		CN19
8	C	CN16
9		CN14
10		CN12
11		CN10
12	D	CN17
13		CN15
14		CN13
15		CN11
16	E	CN8
17		CN6
18		CN4
19		CN2
20	F	CN9
21		CN7
22		CN5
23		CN3

3.8 RC サーボ初期化プログラム：init_servo24ch 関数

```
431 :  void init_servo24ch( void )
432 :  {
433 :          int i;
434 :
435 :          for( i = 0; i < 24; i++ ){
436 :                  pwm_dty[i] = PWM_CYCLE+1;
437 :          }
438 :
439 :          pwm_enb = 1;
440 :
441 :  }
```

RC サーボの制御で使用する変数の初期化を行います。pwm_dty 変数は、PWM 端子を L のままにするため、PWM_CYCLE+1 を入れてコンペアマッチがおこらないようにしています。pwm_enb 変数は、割り込みでの制御を開始するため 1 にします。

3.9 RC サーボ制御プログラム：servo24ch 関数

```
448 :  void servo24ch( int ch, int angle )
449 :  {
450 :          int i;
451 :
452 :          if( 0 <= ch && ch < 24 ){
453 :                  i = SERVO_CENTER + SERVO_STEP * angle;
454 :                  if( 0 <= i && i < PWM_CYCLE ){
455 :                          pwm_dty[ch] = i;
456 :                  }
457 :          }
458 :
459 :  }
```

SERVO_CENTER＋SERVO_STEP×angle

の計算を行い、角度をデューティに変換し pwm_dty 変数に格納しています。

3.10 メインプログラム：main 関数

main 関数は、スタートアップルーチンから呼び出され、最初に実行される C 言語のプログラムです。

プログラム

```
 66 :  void main(void)
 67 :  {
 68 :          int     i;
 69 :
 70 :          // 初期化
 71 :          init();
 72 :
 73 :          init_servo24ch();
 74 :
 75 :          // 起動音
 76 :          beep(Def_500Hz);
 77 :          timer(100);
 78 :          beep(Def_1000Hz);
 79 :          timer(100);
 80 :          beep(0);
 81 :
 82 :          while( pushsw() == 0 ){
 83 :          }
 84 :
 85 :          while(1){
 86 :
 87 :  //              servo24ch( 0, 90 );
 88 :  //              servo24ch( 1, 90 );
 89 :  //              servo24ch( 2, 90 );
 90 :  //              servo24ch( 3, 90 );
 91 :  //              servo24ch( 4, 90 );
 92 :  //              servo24ch( 5, 90 );
 93 :  //              servo24ch( 6, 90 );
 94 :  //              servo24ch( 7, 90 );
 95 :
 96 :  //              servo24ch( 8, 90 );
 97 :  //              servo24ch( 9, 90 );
 98 :  //              servo24ch( 10, 90 );
 99 :  //              servo24ch( 11, 90 );
100 :  //              servo24ch( 12, 90 );
101 :  //              servo24ch( 13, 90 );
102 :  //              servo24ch( 14, 90 );
103 :  //              servo24ch( 15, 90 );
104 :
105 :                  servo24ch( 16, 90 );
106 :                  servo24ch( 17, 90 );
107 :                  servo24ch( 18, 90 );
108 :                  servo24ch( 19, 90 );
109 :                  servo24ch( 20, 90 );
110 :                  servo24ch( 21, 90 );
111 :                  servo24ch( 22, 90 );
112 :                  servo24ch( 23, 90 );
113 :
114 :                  motor( 100, 100 );
115 :
116 :                  timer(1000);
117 :
118 :  //              servo24ch( 0, -90 );
119 :  //              servo24ch( 1, -90 );
120 :  //              servo24ch( 2, -90 );
121 :  //              servo24ch( 3, -90 );
122 :  //              servo24ch( 4, -90 );
123 :  //              servo24ch( 5, -90 );
124 :  //              servo24ch( 6, -90 );
125 :  //              servo24ch( 7, -90 );
```

```
126 :
127 :  //              servo24ch( 8, -90 );
128 :  //              servo24ch( 9, -90 );
129 :  //              servo24ch( 10, -90 );
130 :  //              servo24ch( 11, -90 );
131 :  //              servo24ch( 12, -90 );
132 :  //              servo24ch( 13, -90 );
133 :  //              servo24ch( 14, -90 );
134 :  //              servo24ch( 15, -90 );
135 :
136 :                  servo24ch( 16, -90 );
137 :                  servo24ch( 17, -90 );
138 :                  servo24ch( 18, -90 );
139 :                  servo24ch( 19, -90 );
140 :                  servo24ch( 20, -90 );
141 :                  servo24ch( 21, -90 );
142 :                  servo24ch( 22, -90 );
143 :                  servo24ch( 23, -90 );
144 :
145 :                  motor( -100, -100 );
146 :
147 :                  timer(1000);
148 :
149 :          }
150 :
151 :  }
```

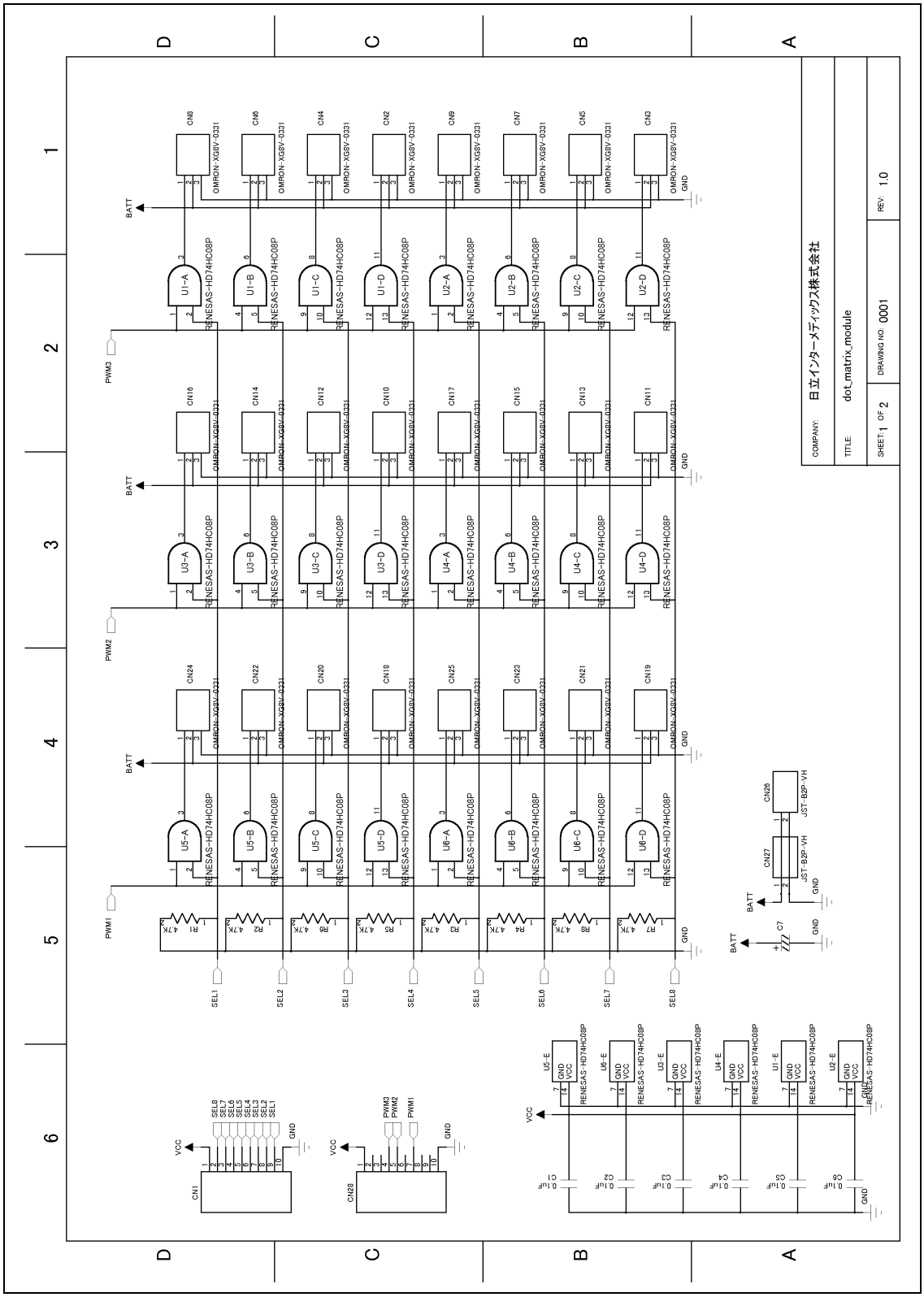
main 関数では、RC サーボの+90°-90°と車体の前進後退を 1 秒ごとに繰り返す処理をしています。

4. 仕様

4.1 仕様

内容	詳細
電源	DC+4.8～6V ※使用する RC サーボの電圧に合わせてください。
RC サーボ	24 個まで制御可能（モータードライバ部分を切り離した場合）
I/O	・RC サーボ 24ch 制御信号入力用コネクタ×2 個 ・RC サーボコネクタ×24 個

4.2 回路図


COMPANY: 日立インターメディックス株式会社
TITLE: dot_matrix_module
DRAWING NO: 0001
REV: 1.0
SHEET 1 OF 2
BATT
GND
VCC
PWM1
PWM2
PWM3
SEL1
SEL2
SEL3
SEL4
SEL5
SEL6
SEL7
SEL8
CN1
CN28
CN26
CN27
JST-B2P-VH
OMRON-XG8V-0331
RENESAS-HD74HC08P

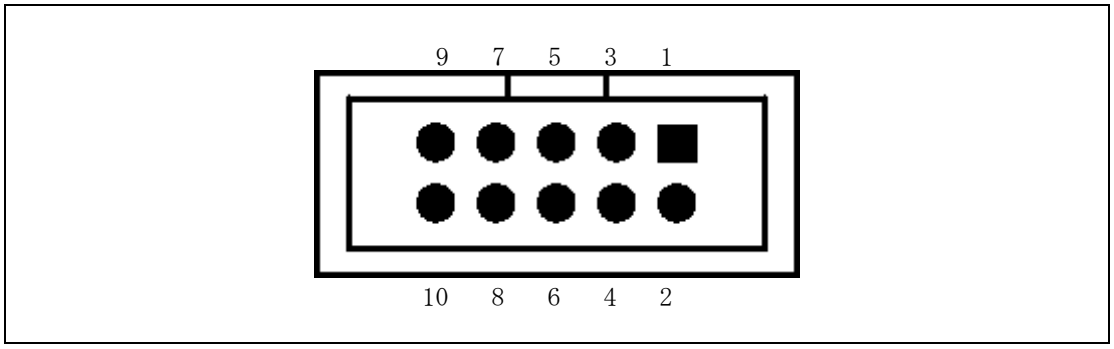
4.3 ポート表

コネクタ	番号	端子名	機能
CN1	1	VCC	
	2		RC サーボ用選択信号 8
	3		RC サーボ用選択信号 7
	4		RC サーボ用選択信号 6
	5		RC サーボ用選択信号 5
	6		RC サーボ用選択信号 4
	7		RC サーボ用選択信号 3
	8		RC サーボ用選択信号 2
	9		RC サーボ用選択信号 1
	10	GND	

コネクタ	番号	端子名	機能
CN28	1	VCC	
	2		
	3		
	4		RC サーボ用 PWM3（E、F グループ）
	5		RC サーボ用 PWM2（C、D グループ）
	6		
	7		RC サーボ用 PWM1（A、B グループ）
	8		
	9		
	10	GND	

4.4 ピン配置図

コネクタ



RC サーボ