23. モータの制御（プロジェクト：motor）

23.1 概要

本章では、ミニマイコンカーVer.2 の左モータ、右モータの回転を制御する方法を紹介します。プログラムで、左モータ、右モータそれぞれを正転、停止、逆転させることができます。また、正転、逆転は、ゆっくり正転、速めに逆転など、プログラムでスピードを制御することができます。スピード制御は、タイマ RD によるリセット同期 PWM モードを使用します。

なお本章では、タイマ RB による割り込みを使っていますがここでは説明していません。タイマ RB による割り込みについては、「**14. 割り込みによるタイマ（プロジェクト：timer2）**」を参照してください。

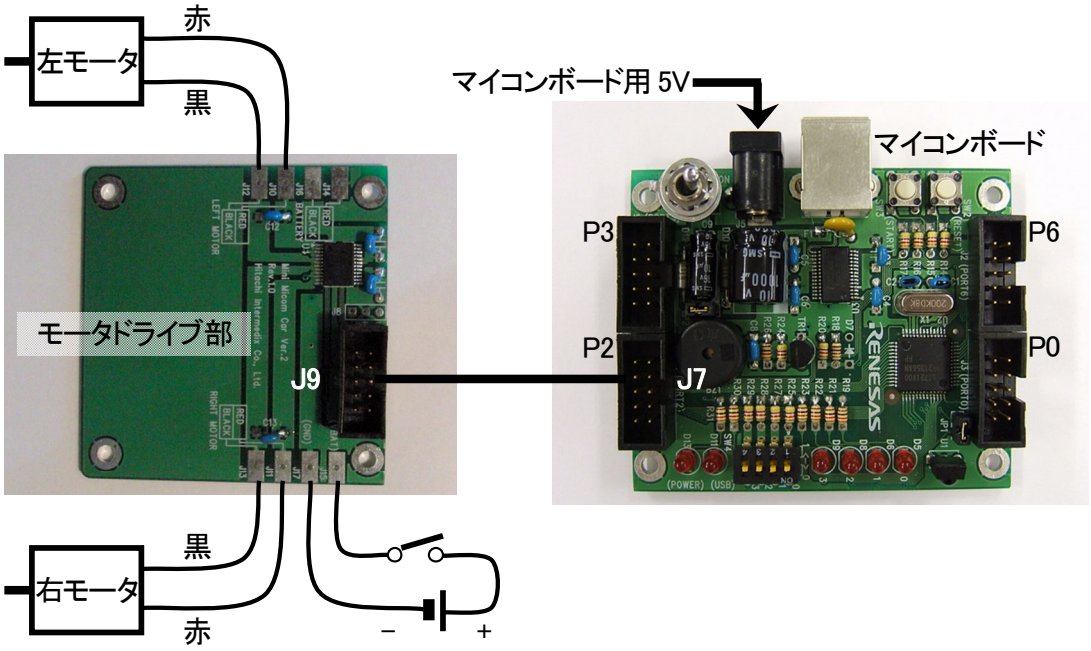
23.2 接続

■使用ポート

マイコンのポート	接続内容
P2 (J7)	ミニマイコンカーVer.2 のモータドライブ部を使用

■接続例

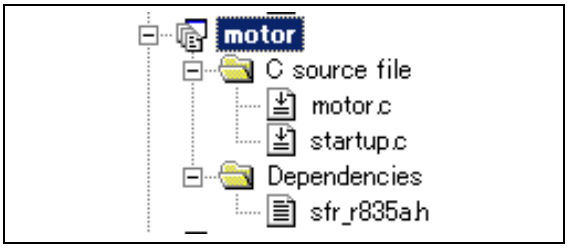
ミニマイコンカーVer.2 のマイコンボードとモータドライブ部を分離している場合の接続図を下記に示します。分離していない（購入時のまま）場合は、特に結線はありません。そのままの状態で、本実習の演習ができます。



■操作方法

操作は特にありません。電源を入れると左右のモータが動き出します。右モータ、左モータの動きをよく観察してください。モータ、タイヤが回りますのでミニマイコンカーVer.2 は浮かした状態で実験してください。

23.3 プロジェクトの構成



	ファイル名	内容
1	startup.c	固定割り込みベクタアドレスの設定、スタートアッププログラム、RAM の初期化（初期値のないグローバル変数、初期値のあるグローバル変数の設定）などを行います。このファイルは共通で、どのプロジェクトもこのファイルから実行されます。
2	motor.c	実際に制御するプログラムが書かれています。R8C/35A の内蔵周辺機能(SFR)の初期化も行います。
3	sfr_r835a.h	R8C/35A マイコンの内蔵周辺機能を制御するためのレジスタ（Special Function Registers)を定義したファイルです。

23.4 プログラム「motor.c」

```
/************************************************************************/
/* 対象マイコン R8C/35A                                                 */
/* ﾌｧｲﾙ内容     ミニマイコンカーVer.2のモータ制御                       */
/* バージョン   Ver.1.20                                                */
/* Date         2010.04.19                                              */
/* Copyright    ルネサスマイコンカーラリー事務局                        */
/*              日立インターメディックス株式会社                        */
/************************************************************************/
/*
入力：マイコンボード上のディップスイッチ
出力：ミニマイコンカーVer.2の右モータ、左モータ

ミニマイコンカーVer.2の右モータ、左モータを制御します。。
マイコンボード上のディップスイッチで、スピード調整することができます。
*/

/*====================================*/
/* インクルード                       */
/*====================================*/
#include "sfr_r835a.h"                  /* R8C/35A SFRの定義ファイル    */

/*====================================*/
/* シンボル定義                       */
/*====================================*/
#define PWM_CYCLE   39999               /* モータPWMの周期              */

/*====================================*/
/* プロトタイプ宣言                   */
/*====================================*/
void init( void );
void timer( unsigned long timer_set );
void motor( int data1, int data2 );
unsigned char dipsw_get( void );

/*====================================*/
/* グローバル変数の宣言               */
/*====================================*/
unsigned long cnt_rb;                   /* タイマRB用                   */

/************************************************************************/
```

```
 41 :  /* メインプログラム                                                      */
 42 :  /************************************************************************/
 43 :  void main( void )
 44 :  {
 45 :      init();                             /* 初期化                       */
 46 :      asm(" fset I ");                    /* 全体の割り込み許可           */
 47 :
 48 :      while( 1 ) {
 49 :          motor( 100 , 0);
 50 :          timer( 1000 );
 51 :          motor( 0, 80 );
 52 :          timer( 1000 );
 53 :          motor( -60, 0 );
 54 :          timer( 1000 );
 55 :          motor( 0, -40 );
 56 :          timer( 1000 );
 57 :          motor( 0, 0 );
 58 :          timer( 1000 );
 59 :      }
 60 :  }
 61 :
 62 :  /************************************************************************/
 63 :  /* R8C/35A スペシャルファンクションレジスタ(SFR)の初期化                */
 64 :  /************************************************************************/
 65 :  void init( void )
 66 :  {
 67 :      int i;
 68 :
 69 :      /* クロックをXINクロック(20MHz)に変更 */
 70 :      prc0  = 1;                          /* プロテクト解除               */
 71 :      cm13  = 1;                          /* P4_6,P4_7をXIN-XOUT端子にする*/
 72 :      cm05  = 0;                          /* XINクロック発振              */
 73 :      for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約10ms) */
 74 :      ocd2  = 0;                          /* システムクロックをXINにする  */
 75 :      prc0  = 0;                          /* プロテクトON                 */
 76 :
 77 :      /* ポートの入出力設定 */
 78 :      prc2 = 1;                           /* PD0のプロテクト解除          */
 79 :      pd0 = 0xe0;                         /* 7-5:LED 4:MicroSW 3-0:Sensor */
 80 :      p1  = 0x0f;                         /* 3-0:LEDは消灯                */
 81 :      pd1 = 0xdf;                         /* 5:RXD0 4:TXD0 3-0:LED        */
 82 :      pd2 = 0xfe;                         /* 7-1:モータドライブ部 0:PushSW*/
 83 :      pd3 = 0xfb;                         /* 4:Buzzer 2:IR                */
 84 :      pd4 = 0x80;                         /* 7:XOUT 6:XIN 5-3:DIP SW 2:VREF*/
 85 :      pd5 = 0x40;                         /* 7:DIP SW                     */
 86 :      pd6 = 0xff;
 87 :
 88 :      /* タイマRBの設定 */
 89 :      /* 割り込み周期 = 1 / 20[MHz]   * (TRBPRE+1) * (TRBPR+1)
 90 :                      = 1 / (20*10^6) * 200        * 100
 91 :                      = 0.001[s] = 1[ms]
 92 :      */
 93 :      trbmr  = 0x00;                      /* 動作モード、分周比設定       */
 94 :      trbpre = 200-1;                     /* プリスケーラレジスタ         */
 95 :      trbpr  = 100-1;                     /* プライマリレジスタ           */
 96 :      trbic  = 0x07;                      /* 割り込み優先レベル設定       */
 97 :      trbcr  = 0x01;                      /* カウント開始                 */
 98 :
 99 :      /* タイマRD リセット同期PWMモードの設定*/
100 :      /* PWM周期 = 1 / 20[MHz]   * カウントソース * (TRDGRA0+1)
101 :                 = 1 / (20*10^6) * 8              * 40000
102 :                 = 0.016[s] = 16[ms]
103 :      */
104 :      trdfcr  = 0x01;                     /* リセット同期PWMモードに設定  */
105 :      trdmr   = 0xf0;                     /* バッファレジスタ設定         */
106 :      trdoer1 = 0xcd;                     /* 出力端子の選択               */
107 :      trdpsr0 = 0x08;                     /* TRDIOB0,C0,D0端子設定        */
108 :      trdpsr1 = 0x05;                     /* TRDIOA1,B1,C1,D1端子設定     */
109 :      trdcr0  = 0x23;                     /* ソースカウントの選択:f8      */
110 :      trdgra0 = trdgrc0 = PWM_CYCLE;      /* 周期                         */
111 :      trdgrb0 = trdgrd0 = 0;              /* P2_2端子のON幅設定           */
112 :      trdgra1 = trdgrc1 = 0;              /* P2_4端子のON幅設定           */
113 :      trdgrb1 = trdgrd1 = 0;              /* P2_5端子のON幅設定           */
114 :      trdstr  = 0x0d;                     /* TRD0カウント開始             */
115 :  }
116 :
```

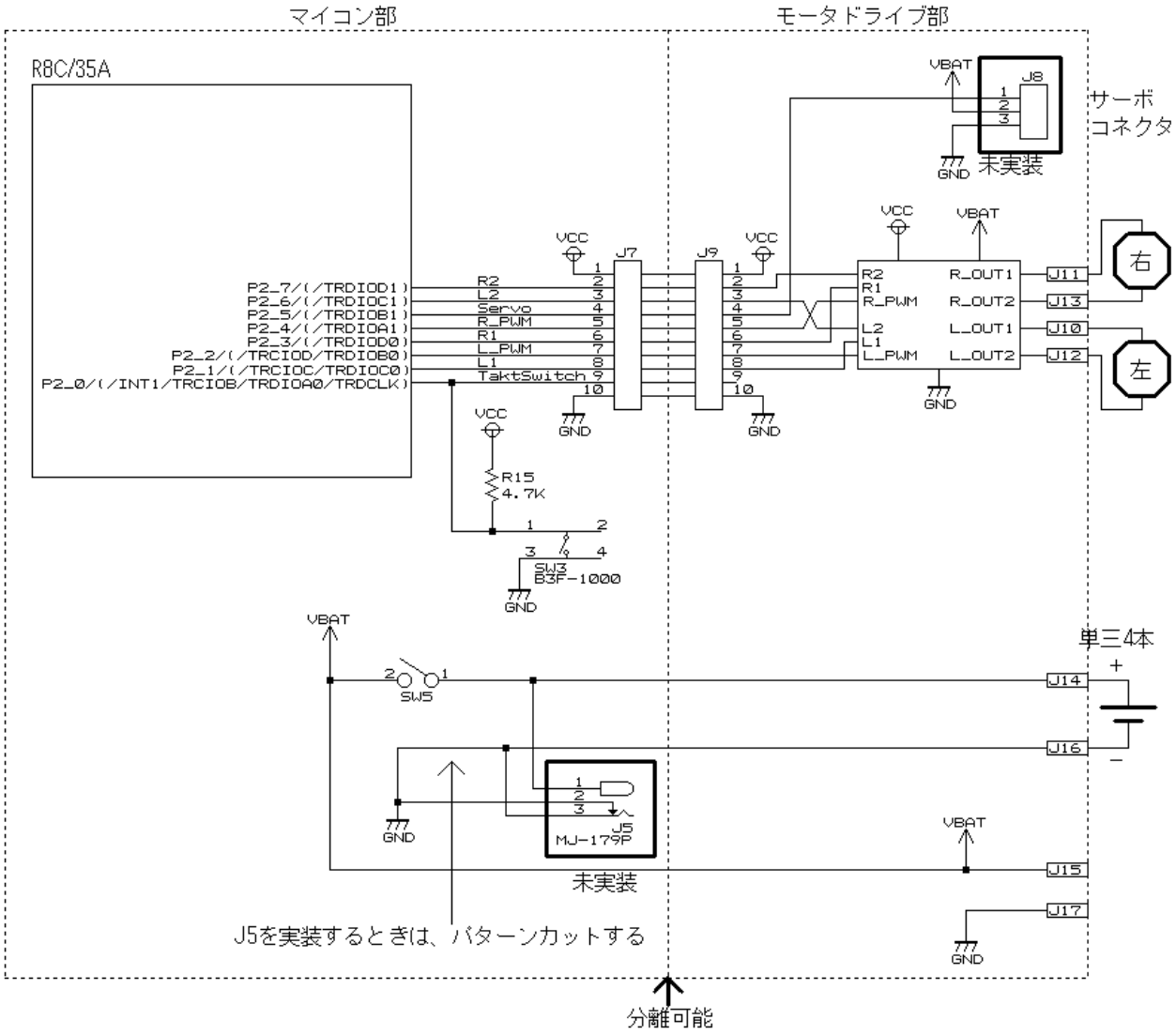
```
117 :  /************************************************************************/
118 :  /* ディップスイッチ値読み込み                                           */
119 :  /* 戻り値 スイッチ値 0～15                                              */
120 :  /************************************************************************/
121 :  unsigned char dipsw_get( void )
122 :  {
123 :      unsigned char sw, sw1, sw2;
124 :
125 :      sw1 = (p5>>4) & 0x08;               /* ディップスイッチ読み込み3    */
126 :      sw2 = (p4>>3) & 0x07;               /* ディップスイッチ読み込み2,1,0*/
127 :      sw = sw1 | sw2;                     /* P5とP4の値を合わせる         */
128 :
129 :      return  sw;
130 :  }
131 :
132 :  /************************************************************************/
133 :  /* タイマ本体                                                           */
134 :  /* 引数    タイマ値 1=1ms                                              */
135 :  /************************************************************************/
136 :  void timer( unsigned long timer_set )
137 :  {
138 :      cnt_rb = 0;
139 :      while( cnt_rb < timer_set );
140 :  }
141 :
142 :  /************************************************************************/
143 :  /* タイマRB 割り込み処理                                                */
144 :  /************************************************************************/
145 :  #pragma interrupt intTRB(vect=24)
146 :  void intTRB( void )
147 :  {
148 :      cnt_rb++;
149 :  }
150 :
151 :  /************************************************************************/
152 :  /* モータ速度制御                                                       */
153 :  /* 引数    左モータ:-100～100、右モータ:-100～100                       */
154 :  /*         0で停止、100で正転100%、-100で逆転100%                       */
155 :  /* 戻り値 なし                                                          */
156 :  /************************************************************************/
157 :  void motor( int data1, int data2 )
158 :  {
159 :      int    motor_r, motor_l, sw_data;
160 :
161 :      sw_data = dipsw_get() + 5;
162 :      motor_l = data1 * sw_data / 20;
163 :      motor_r = data2 * sw_data / 20;
164 :
165 :      /* 左モータ制御 */
166 :      if( motor_l >= 0 ) {
167 :          p2 &= 0xfd;
168 :          p2 |= 0x40;
169 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
170 :      } else {
171 :          p2 |= 0x02;
172 :          p2 &= 0xbf;
173 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
174 :      }
175 :
176 :      /* 右モータ制御 */
177 :      if( motor_r >= 0 ) {
178 :          p2 &= 0xf7;
179 :          p2 |= 0x80;
180 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
181 :      } else {
182 :          p2 |= 0x08;
183 :          p2 &= 0x7f;
184 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
185 :      }
186 :  }
187 :
188 :  /************************************************************************/
189 :  /* end of file                                                          */
190 :  /************************************************************************/
```

23.5 モータドライブ部

23.5.1 マイコン部とモータドライブ部の接続

モータドライブ部の J9 は、マイコン部の J7 に接続します。基板を分離していない場合は、あらかじめパターンで接続されているので何もする必要はありません。分離した場合は、フラットケーブルなどで J7 と J9 を接続しモータ用電源(VBAT)の配線も行ってください。

マイコン部とモータドライブ部の結線を下記に示します。



※サーボコネクタ(J8)、DC ジャック(J5)はオプションです。

23.5.2 J7 とJ9 の詳細

J7 と J9 のピン番号に対する信号名を下表に示します。J7 と J9 はあらかじめパターンで接続されています。マイコン部とモータドライブ部を分離した場合は、フラットケーブルなどで J7 と J9 を接続してください。

プログラムで P2_2 端子、P2_4 端子を PWM 端子にします。オプションでサーボを取り付けることができます。今回はサーボがありませんが、今後のことも考えて P2_5 端子も PWM 端子にしておきます。他は I/O 端子に設定します。PWM の周期は 16ms にします。

J9			J7（マイコン部）			
ピン番号	信号名		ピン番号	信号名	入出力設定	説明
1	VCC(+5V)	―	1	VCC(+5V)		
2	モータ右 2	―	2	P2_7	出力	P2_7 は通常の I/O ポートです。
3	モータ左 2	―	3	P2_6	出力	P2_6 は通常の I/O ポートです。
4	**サーボ（オプション）**	―	4	P2_5	出力（PWM 波形出力）	この端子は PWM 出力許可にします。TRDGRD1 で ON 幅を設定します。
5	**モータ右 PWM**	―	5	P2_4	出力（PWM 波形出力）	この端子は PWM 出力許可にします。TRDGRC1 で ON 幅を設定します。
6	モータ右 1	―	6	P2_3	出力	P2_3 は通常の I/O ポートです。
7	**モータ左 PWM**	―	7	P2_2	出力（PWM 波形出力）	この端子は PWM 出力許可にします。TRDGRD0 で ON 幅を設定します。
8	モータ左 1	―	8	P2_1	出力	P2_1 は通常の I/O ポートです。
9	未接続	―	9	P2_0	入力	P2_0 は通常の I/O ポートです。マイコンボード上のタクトスイッチに繋がっています。
10	GND	―	10	GND		

↑パターン上で接続されています。

23.5.3 左モータの動作

左モータは、P2_1、P2_2、P2_6 の 3 端子で制御します。

モータ左 1 P2_1	モータ左 2 P2_6	モータ左 PWM P2_2	モータ動作
1	1	0 または 1	ブレーキ
0	0	0 または 1	フリー
0	1	PWM	PWM="1"なら正転、"0"ならブレーキ
1	0	PWM	PWM="1"なら逆転、"0"ならブレーキ

左モータを正転させたい場合、P2_1="0"、P2_6="1"にして、P2_2 から PWM 波形を出力すると、左モータが PWM の割合に応じて正転します。例えば、PWM が 0%ならモータの回転は停止、PWM が 50%ならモータの回転は正転 50%、PWM100%ならモータの回転は正転 100%になります。
左モータを逆転させたい場合、P2_1="1"、P2_6="0"にして、P2_2 から PWM 波形を出力すると、左モータが PWM の割合に応じて逆転します。例えば、PWM が 0%ならモータの回転は停止、PWM が 50%ならモータの回転は逆転 50%、PWM100%ならモータの回転は逆転 100%になります。

23.5.4 右モータの動作

右モータは、P2_3、P2_4、P2_7 の 3 端子で制御します。

モータ右 1 P2_3	モータ右 2 P2_7	モータ右 PWM P2_4	モータ動作
1	1	0 または 1	ブレーキ
0	0	0 または 1	フリー
0	1	PWM	PWM="1"なら正転、"0"ならブレーキ
1	0	PWM	PWM="1"なら逆転、"0"ならブレーキ

右モータを正転させたい場合、P2_3="0"、P2_7="1"にして、P2_4 から PWM 波形を出力すると、右モータが PWM の割合に応じて正転します。例えば、PWM が 0%ならモータの回転は停止、PWM が 50%ならモータの回転は正転 50%、PWM100%ならモータの回転は正転 100%になります。
右モータを逆転させたい場合、P2_3="1"、P2_7="0"にして、P2_4 から PWM 波形を出力すると、右モータが PWM の割合に応じて逆転します。例えば、PWM が 0%ならモータの回転は停止、PWM が 50%ならモータの回転は逆転 50%、PWM100%ならモータの回転は逆転 100%になります。

23.6 プログラムの解説

23.6.1 init関数(タイマRDの設定)

タイマ RD を使いリセット同期 PWM モードの設定を行います。

```
 22 :  /*====================================*/
 23 :  /* シンボル定義                       */
 24 :  /*====================================*/
 25 :  #define PWM_CYCLE   39999              /* モータPWMの周期              */

中略

104 :      trdfcr  = 0x01;                    /* リセット同期PWMモードに設定  */
105 :      trdmr   = 0xf0;                    /* バッファレジスタ設定         */
106 :      trdoer1 = 0xcd;                    /* 出力端子の選択               */
107 :      trdpsr0 = 0x08;                    /* TRDIOB0,C0,D0端子設定        */
108 :      trdpsr1 = 0x05;                    /* TRDIOA1,B1,C1,D1端子設定     */
109 :      trdcr0  = 0x23;                    /* ソースカウントの選択:f8      */
110 :      trdgra0 = trdgrc0 = PWM_CYCLE;     /* 周期                         */
111 :      trdgrb0 = trdgrd0 = 0;             /* P2_2端子のON幅設定           */
112 :      trdgra1 = trdgrc1 = 0;             /* P2_4端子のON幅設定           */
113 :      trdgrb1 = trdgrd1 = 0;             /* P2_5端子のON幅設定           */
114 :      trdstr  = 0x0d;                    /* TRD0カウント開始             */
```

PWM の周期は 16ms です。110 行で、タイマ RD ジェネラルレジスタ A0(TRDGRA0)、タイマ RD ジェネラルレジスタ C0(TRDGRC0)に計算結果である 39999 を設定します。今回は 25 行目で「PWM_CYCLE」という PWM の周期を設定する定数を定義して、「PWM_CYCLE」に 39999 を割り当てます。周期を変更する場合は、25 行にある「39999」を変更してください。

①タイマ RD 機能制御レジスタ(TRDFCR:Timer RD function control register)の設定

タイマ RD の機能を設定します。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7		"0"を設定	0
bit6	外部クロック入力選択ビット stclk_trdfcr	0:外部クロック入力無効 1:外部クロック入力有効 今回は、内部のクロックを使用しますので、無効を選択します。	0
bit5,4		"00"を設定	00
bit3	逆相出力レベル選択ビット (リセット同期 PWM モードまたは相補 PWM モード時) ols1_trdfcr	0:初期出力"H"、アクティブレベル"L" 1:初期出力"L"、アクティブレベル"H" 今回は、"0"を設定します。	0
bit2	正相出力レベル選択ビット (リセット同期 PWM モードまたは相補 PWM モード時) ols0_trdfcr	0:初期出力"H"、アクティブレベル"L" 1:初期出力"L"、アクティブレベル"H" 今回は、"0"を設定します。	0
bit1,0	コンビネーションモード選択ビット bit1:cmd1_trdfcr bit0:cmd0_trdfcr	リセット同期 PWM モードでは"01"(リセット同期 PWM モード)にしてください	01

タイマ RD 機能制御レジスタ(TRDFCR)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	0	0	0	0	**0**	**0**	**0**	**1**
16 進数	0				1			

②タイマ RD モードレジスタ(TRDMR:Timer RD mode register)の設定

タイマ RD のジェネラルレジスタをバッファレジスタとして使用するかどうか設定します。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7	TRDGRD1 レジスタ機能選択ビット bfd1_trdmr	0:ジェネラルレジスタ 1:TRDGRB1 レジスタのバッファレジスタ バッファレジスタとして使用します。	1
bit6	TRDGRC1 レジスタ機能選択ビット bfc1_trdmr	0:ジェネラルレジスタ 1:TRDGRA1 レジスタのバッファレジスタ バッファレジスタとして使用します。	1
bit5	TRDGRD0 レジスタ機能選択ビット bfd0_trdmr	0:ジェネラルレジスタ 1:TRDGRB0 レジスタのバッファレジスタ バッファレジスタとして使用します。	1
bit4	TRDGRC0 レジスタ機能選択ビット bfc0_trdmr	0:ジェネラルレジスタ 1:TRDGRA0 レジスタのバッファレジスタ バッファレジスタとして使用します。	1
bit3～0		″0000″を設定	0000

タイマ RD モードレジスタ(TRDMR)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	1	1	1	1	0	0	0	0
16 進数	F				0			

③タイマ RD アウトプットマスタ許可レジスタ 1(TRDOER1：Timer RD output master enable register 1)の設定

リセット同期 PWM モードは 7 端子から PWM 波形を出力することができますが、出力するか、通常の I/O ポートとして使用するかを選択できます。
今回は、P2_2 端子、P2_4 端子、P2_5 端子を PWM 波形出力、その他の端子は I/O ポートとして使用します。

設定 bit	上：ビット名 下：シンボル	内容	今回の内容
bit7	TRDIOD1(P2_7)出力禁止ビット ed1_trdoer1	0：出力許可 1：出力禁止(TRDIOD1 端子はプログラマブル入出力ポート) 今回は、出力を禁止します。	1
bit6	TRDIOC1(P2_6)出力禁止ビット ec1_trdoer1	0：出力許可 1：出力禁止(TRDIOC1 端子はプログラマブル入出力ポート) 今回は、出力を禁止します。	1
bit5	TRDIOB1(P2_5)出力禁止ビット eb1_trdoer1	0：出力許可 1：出力禁止(TRDIOB1 端子はプログラマブル入出力ポート) サーボ PWM 用に、出力を許可します。	0
bit4	TRDIOA1(P2_4)出力禁止ビット ea1_trdoer1	0：出力許可 1：出力禁止(TRDIOA1 端子はプログラマブル入出力ポート) 右モータ PWM 用に、出力を許可します。	0
bit3	TRDIOD0(P2_3)出力禁止ビット ed0_trdoer1	0：出力許可 1：出力禁止(TRDIOD0 端子はプログラマブル入出力ポート) 今回は、出力を禁止します。	1
bit2	TRDIOC0(P2_1)出力禁止ビット ec0_trdoer1	0：出力許可 1：出力禁止(TRDIOC0 端子はプログラマブル入出力ポート) 今回は、出力を禁止します。	1
bit1	TRDIOB0(P2_2)出力禁止ビット eb0_trdoer1	0：出力許可 1：出力禁止(TRDIOB0 端子はプログラマブル入出力ポート) 左モータ PWM 用に、出力を許可します。	0
bit0		″1″を設定	1

タイマ RD アウトプットマスタ許可レジスタ 1(trdoer1)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	**1**	**1**	**0**	**0**	**1**	**1**	**0**	1
16 進数	c				d			

④タイマ RD 端子選択レジスタ 0(TRDPSR0:Timer RD function select register 0)の設定

PWM 波形の出力端子をどのポートに割り当てるか設定します。タイマ RD 端子選択レジスタ 0(TRDPSR0)では、TRDIOD0 端子、TRDIOC0 端子、TRDIOB0 端子の割り当てを設定します。

R8C/35A では、割り当てる端子は決まっており端子を変更することができません。PWM 端子として割り当てるか、割り当てないか(通常の I/O ポートとして使用するか)を設定します。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7		"0"を設定	0
bit6	TRDIOD0 端子選択ビット trdiod0sel0	0:TRDIOD0 端子は使用しない 1:P2_3 に割り当てる 今回は、使用しません。	0
bit5,4	TRDIOC0 端子選択ビット bit5:trdioc0sel1 bit4:trdioc0sel0	00:TRDIOC0 端子は使用しない 01:設定しないでください 10:P2_1 に割り当てる 11:設定しないでください 今回は、使用しません。	00
bit3,2	TRDIOB0 端子選択ビット bit3:trdiob0sel1 bit2:trdiob0sel0	00:TRDIOB0 端子は使用しない 01:設定しないでください 10:P2_2 に割り当てる 11:設定しないでください 今回は、P2_2 に割り当てて、この端子から PWM 波形を出力します。左モータ PWM 用です。	10
bit1,0		"0"を設定	00

タイマ RD 端子選択レジスタ 0(TRDPSR0)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	0	**0**	**0**	**0**	**1**	**0**	0	0
16 進数	0				8			

⑤タイマ RD 端子選択レジスタ 1(TRDPSR1:Timer RD function select register 1)の設定

PWM 波形の出力端子をどのポートに割り当てるか設定します。タイマ RD 端子選択レジスタ 1(TRDPSR1)では、TRDIOD1 端子、TRDIOC1 端子、TRDIOB1 端子、TRDIOA1 端子の割り当てを設定します。
R8C/35A では、割り当てる端子は決まっており端子を変更することができません。PWM 端子として割り当てるか、割り当てないか(通常の I/O ポートとして使用するか)を設定します。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7		"0"を設定	0
bit6	TRDIOD1 端子選択ビット trdiod1sel0	0:TRDIOD1 端子は使用しない 1:P2_7 に割り当てる 今回は、使用しません。	0
bit5		"0"を設定	0
bit4	TRDIOC1 端子選択ビット trdioc1sel0	0:TRDIOC1 端子は使用しない 1:P2_6 に割り当てる 今回は、使用しません。	0
bit3		"0"を設定	0
bit2	TRDIOB1 端子選択ビット trdiob1sel0	0:TRDIOB1 端子は使用しない 1:P2_5 に割り当てる 今回は、P2_5 に割り当てて、この端子から PWM 波形を出力します。サーボ PWM 用です。	1
bit1		"0"を設定	0
bit0	TRDIOA1 端子選択ビット trdioa1sel0	0:TRDIOA1 端子は使用しない 1:P2_4 に割り当てる 今回は、P2_4 に割り当てて、この端子から PWM 波形を出力します。右モータ PWM 用です。	1

タイマ RD 端子選択レジスタ 1(TRDPSR1)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	0	**0**	0	**0**	0	**1**	0	**1**
16 進数	0				5			

⑥タイマ RD 制御レジスタ 0(TRDCR0：Timer RD control register 0)の設定

タイマ RD カウンタ 0（TRD0）がカウントアップする時間などを選択します。

設定 bit	上：ビット名 下：シンボル	内容	今回の内容
bit7～5	TRD0 カウンタクリア選択ビット bit7:cclr2_trdcr0 bit6:cclr1_trdcr0 bit5:cclr0_trdcr0	リセット同期 PWM モードの場合は、"001"を設定してください (TRDGRA0 とのコンペア一致で TRD0 レジスタクリアにしてください)	001
bit4,3	外部クロックエッジ選択ビット (注 3) bit4:ckeg1_trdcr0 bit3:ckeg0_trdcr0	00：立ち上がりエッジでカウント 01：立ち下がりエッジでカウント 10：両エッジでカウント 11：設定しないでください 外部クロックは使いませんので何を設定しても変化ありません。今回は 00 を設定します。	00
bit2～0	カウントソース選択ビット bit2:tck2_trdcr0 bit1:tck1_trdcr0 bit0:tck0_trdcr0	000：f1 (1/20MHz＝50ns) 001：f2 (2/20MHz＝100ns) 010：f4 (4/20MHz＝200ns) 011：f8 (8/20MHz＝400ns) 100：f32 (32/20MHz＝1600ns) 101：TRDCLK 入力(注 1)または fC2 (注 2) fC2 = 2/XCIN クロック＝今回は未接続 110：fOCO40M（高速オンチップオシレータ 40MHz＝今回は未接続) 111：fOCO-F(注 4)（高速オンチップオシレータを FRA2 で分周したクロック＝今回は未接続) タイマ RD カウンタ 0(TRD0)がカウントアップする時間を設定します。今回は"011"を設定します。TRD0 は、400ns ごとに+1 していきます。	011

注 1. TRDECR レジスタの ITCLKi ビットが“0”(TRDCLK 入力)かつ TRDFCR レジスタの STCLK ビットが“1”(外部クロック入力有効)のとき、有効です。
注 2. タイマモードで、TRDECR レジスタの ITCLKi ビットが“1”(fC2)のとき有効です。
注 3. TCK2～TCK0 ビットが“101”(TRDCLK 入力または fC2)、TRDECR レジスタの ITCLKi ビットが“0”(TRDCLK 入力)、かつ TRDFCR レジスタの STCLK ビットが“1”(外部クロック入力有効)のとき、有効です。
注 4. fOCO-F を選択するとき、CPU クロックより速いクロック周波数に fOCO-F を設定してください。

タイマ RD 制御レジスタ 0(TRDCR0)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	0	0	1	0	0	**0**	**1**	**1**
16 進数	2				3			

※タイマ RD 制御レジスタ 0(TRDCR0)カウントソース選択ビットの設定方法

タイマ RD 制御レジスタ 0(TRDCR0)のカウントソース選択ビット(bit2～0)で、タイマ RD カウンタ 0(TRD0)がどのくらいの間隔で+1 するか設定します。TRD0 は、0 からスタートして最大 65,535 までカウントアップします。65,535 の次は 0 に戻ります。PWM の周期や ON 幅は TRD0 の値を基準にするので、カウントアップする時間×65,536 以上の時間を設定することができません。

タイマ RD 制御レジスタ 0(TRDCR0)のカウントソース選択ビットの値と、周期の関係を下記に示します。

bit2～0	内容
000	タイマ RD カウンタ 0(TRD0)がカウントアップする時間を、f1 に設定します。時間は、 f1/20MHz＝1/20MHz＝50ns 設定できる PWM 周期の最大は、 50ns×65,536＝**3.2768ms** よって、この時間以内の PWM 周期を設定する場合は"000"を設定、これ以上の PWM 周期を設定したい場合は次以降の値を検討します。
001	タイマ RD カウンタ 0(TRD0)がカウントアップする時間を、f2 に設定します。時間は、 f2/20MHz＝2/20MHz＝100ns 設定できる PWM 周期の最大は、 100ns×65,536＝**6.5536ms** よって、この時間以内の PWM 周期を設定する場合は"001"を設定、これ以上の PWM 周期を設定したい場合は次以降の値を検討します。
010	タイマ RD カウンタ 0(TRD0)がカウントアップする時間を、f4 に設定します。時間は、 f4/20MHz＝4/20MHz＝200ns 設定できる PWM 周期の最大は、 200ns×65,536＝**13.1072ms** よって、この時間以内の PWM 周期を設定する場合は"010"を設定、これ以上の PWM 周期を設定したい場合は次以降の値を検討します。
011	タイマ RD カウンタ 0(TRD0)がカウントアップする時間を、f8 に設定します。時間は、 f8/20MHz＝8/20MHz＝400ns 設定できる PWM 周期の最大は、 400ns×65,536＝**26.2144ms** よって、この時間以内の PWM 周期を設定する場合は"011"を設定、これ以上の PWM 周期を設定したい場合は次以降の値を検討します。
100	タイマ RD カウンタ 0(TRD0)がカウントアップする時間を、f32 に設定します。時間は、 f32/20MHz＝32/20MHz＝1600ns 設定できる PWM 周期の最大は、 1600ns×65,536＝**104.8576ms** よって、この時間以内の PWM 周期を設定する場合は"100"を設定します。**これ以上の PWM 周期を設定することはできません**。これ以上の PWM 周期を設定しなくても良いように、回路側を工夫してください。

今回は、PWM 波形の周期を 16ms にするので、

①"000"の設定…最大の PWM 周期は 3.2768ms、今回設定したい 16ms の周期を設定できないので不可

②"001"の設定…最大の PWM 周期は 6.5536ms、今回設定したい 16ms の周期を設定できないので不可

③"010"の設定…最大の PWM 周期は 13.1072ms、今回設定したい 16ms の周期を設定できないので不可

④"011"の設定…最大の PWM 周期は 26.2144ms、今回設定したい 16ms の周期を設定できるので OK

よって、"011"を設定します。

⑦タイマ RD ジェネラルレジスタ A0(TRDGRA0：Timer RD General register A0)、
タイマ RD ジェネラルレジスタ C0(TRDGRC0：Timer RD General register C0)の設定

タイマ RD ジェネラルレジスタ A0(TRDGRA0)に値を設定することによって、出力する PWM 波形の周期を設定します。
PWM 周期は、下記の式で決まります。

PWM 周期＝タイマ RD カウンタ 0 のカウントソース×(TRDGRA0＋1)

TRDGRA0 を左辺に移動して、TRDGRA0 を求める式に変形します。

TRDGRA0＝PWM 周期／タイマ RD カウンタ 0 のカウントソースー1

今回、PWM 波形の周期を 16ms にします。タイマ RD カウンタ 0 のカウントソースとは、タイマ RD 制御レジスタ 0(TRDCR0)の bit2～0 で設定した時間のことで、今回は 400ns に設定しています。よって、タイマ RD ジェネラルレジスタ A0(TRDGRA0)は次のようになります。

TRDGRA0＝周期　　／カウントソースー1
TRDGRA0＝$(16\times10^{-3})/(400\times10^{-9})-1$
TRDGRA0＝40000－1 ＝ 39999

TRDGRA0 の値は、65,535 以下にする必要があります。今回の結果は、65,535 以下なので、400ns の設定で大丈夫です。65,536 以上になった場合、タイマ RD 制御レジスタ 0(TRDCR0)の bit2～0 の設定を大きい時間にしてください。

■設定のポイント
※1…タイマ RD ジェネラルレジスタ A0(TRDGRA0)を使うときは、タイマ RD ジェネラルレジスタ C0(TRDGRC0)をバッファレジスタに設定して、ペアで使用してください。
※2…タイマ RD ジェネラルレジスタ A0(TRDGRA0)の設定は、イニシャライズ時に 1 回だけ行ってください。2 回目以降、周期を変更したい場合は、タイマ RD ジェネラルレジスタ C0(TRDGRC0)に値を設定してください。
※3…タイマ RD ジェネラルレジスタ C0(TRDGRC0)のイニシャライズ時に設定する値は、タイマ RD ジェネラルレジスタ A0(TRDGRA0)と同じ値にしてください。

プログラム例を下記に示します。

```
void main( void )
{
    // メインプログラム
    init();                          /* レジスタの初期化 */
    ～～～～～～～～～～～～～～～～
    trdgrc0 = 19999;                 /* 2 回目以降は必ず trdgrc0 に設定する */
    ～～～～～～～～～～～～～～～～
    trdgrc0 = 9999;                  /* 2 回目以降は必ず trdgrc0 に設定する */
}
void init( void )
{
    // レジスタの初期化
    ～～～～～～～～～～～～～～～～
    trdgra0 = trdgrc0 = 39999;       /* 1 回だけ trdgra0 に値を設定     */
                                     /* trdgrc0 にも同じ値を設定する   */
    ～～～～～～～～～～～～～～～～
}
```

⑧タイマ RD ジェネラルレジスタ B0(TRDGRB0：Timer RD General register B0)、
タイマ RD ジェネラルレジスタ D0(TRDGRD0：Timer RD General register D0)の設定

タイマ RD ジェネラルレジスタ B0(TRDGRB0)に値を設定することによって、P2_2 端子から出力される PWM 波形の ON 幅を設定します。P2_3 端子からは、その反転した波形が出力されます。
P2_2 端子から出力される PWM 波形の ON 幅は、下記の式で決まります。

P2_2 端子から出力される PWM 波形の ON 幅＝タイマ RD カウンタ 0 のカウントソース×(TRDGRB0＋1)

TRDGRB0 を左辺に移動して、TRDGRB0 を求める式に変形します。

TRDGRB0＋1＝P2_2 端子から出力される PWM 波形の ON 幅／タイマ RD カウンタ 0 のカウントソース

今回、タイマ RD カウンタ 0 のカウントソースは 400ns です。例えば ON 幅を 1ms にするなら、タイマ RD ジェネラルレジスタ B0(TRDGRB0)の値は次のようになります。

TRDGRB0＋1＝ON 幅　／カウントソース
TRDGRB0＋1＝$(1\times10^{-3})/(400\times10^{-9})$
TRDGRB0＋1＝2500
TRDGRB0　　＝2499

■設定のポイント

※1…タイマ RD ジェネラルレジスタ B0(TRDGRB0)を使うときは、タイマ RD ジェネラルレジスタ D0(TRDGRD0)をバッファレジスタに設定して、ペアで使用してください。
※2…タイマ RD ジェネラルレジスタ B0(TRDGRB0)の設定は、イニシャライズ時に 1 回だけ行ってください。2 回目以降、ON 幅を変更したい場合は、タイマ RD ジェネラルレジスタ D0(TRDGRD0)に値を設定してください。
※3…タイマ RD ジェネラルレジスタ D0(TRDGRD0)のイニシャライズ時に設定する値は、タイマ RD ジェネラルレジスタ B0(TRDGRB0)と同じ値にしてください。

⑨タイマ RD ジェネラルレジスタ A1(TRDGRA1：Timer RD General register A1)、
タイマ RD ジェネラルレジスタ C1(TRDGRC1：Timer RD General register C1)の設定

タイマ RD ジェネラルレジスタ A1(TRDGRA1)に値を設定することによって、P2_4 端子から出力される PWM 波形の ON 幅を設定します。P2_6 端子からは、その反転した波形が出力されます。
P2_4 端子から出力される PWM 波形の ON 幅は、下記の式で決まります。

P2_4 端子から出力される PWM 波形の ON 幅＝タイマ RD カウンタ 0 のカウントソース×(TRDGRA1＋1)

TRDGRA1 を左辺に移動して、TRDGRA1 を求める式に変形します。

TRDGRA1＝P2_4 端子から出力される PWM 波形の ON 幅／タイマ RD カウンタ 0 のカウントソース－1

今回、タイマ RD カウンタ 0 のカウントソースは 400ns です。ON 幅を 10ms にするなら、タイマ RD ジェネラルレジスタ A1(TRDGRA1)は次のようになります。

TRDGRA1＝ON 幅　　／カウントソース－1
TRDGRA1＝$(10\times10^{-3})/(400\times10^{-9})-1$
TRDGRA1＝25000－1　＝　24999

■設定のポイント

※1…タイマ RD ジェネラルレジスタ A1(TRDGRA1)を使うときは、タイマ RD ジェネラルレジスタ C1(TRDGRC1)をバッファレジスタに設定して、ペアで使用してください。
※2…タイマ RD ジェネラルレジスタ A1(TRDGRA1)の設定は、イニシャライズ時に 1 回だけ行ってください。2 回目以降、ON 幅を変更したい場合は、タイマ RD ジェネラルレジスタ C1(TRDGRC1)に値を設定してください。
※3…タイマ RD ジェネラルレジスタ C1(TRDGRC1)のイニシャライズ時に設定する値は、タイマ RD ジェネラルレジスタ A1(TRDGRA1)と同じ値にしてください。

⑩タイマ RD ジェネラルレジスタ B1(TRDGRB1：Timer RD General register B1)、
タイマ RD ジェネラルレジスタ D1(TRDGRD1：Timer RD General register D1)の設定

タイマ RD ジェネラルレジスタ B1(TRDGRB1)に値を設定することによって、P2_5 端子から出力される PWM 波形の ON 幅を設定します。P2_7 端子からは、その反転した波形が出力されます。
P2_5 端子から出力される PWM 波形の ON 幅は、下記の式で決まります。

P2_5 端子から出力される PWM 波形の ON 幅＝タイマ RD カウンタ 0 のカウントソース×(TRDGRB1＋1)

TRDGRB1 を左辺に移動して、TRDGRB1 を求める式に変形します。

TRDGRB1＝P2_5 端子から出力される PWM 波形の ON 幅／タイマ RD カウンタ 0 のカウントソース－1

今回、タイマ RD カウンタ 0 のカウントソースは 400ns です。ON 幅を 15ms にするなら、タイマ RD ジェネラルレジスタ B1(TRDGRB1)は次のようになります。

TRDGRB1＝ON 幅　／カウントソース－1
TRDGRB1＝$(15\times10^{-3})/(400\times10^{-9})-1$
TRDGRB1＝37500－1 ＝ 37499

■設定のポイント

※1…タイマ RD ジェネラルレジスタ B1(TRDGRB1)を使うときは、タイマ RD ジェネラルレジスタ D1(TRDGRD1)をバッファレジスタに設定して、ペアで使用してください。
※2…タイマ RD ジェネラルレジスタ B1(TRDGRB1)の設定は、イニシャライズ時に 1 回だけ行ってください。2 回目以降、ON 幅を変更したい場合は、タイマ RD ジェネラルレジスタ D1(TRDGRD1)に値を設定してください。
※3…タイマ RD ジェネラルレジスタ D1(TRDGRD1)のイニシャライズ時に設定する値は、タイマ RD ジェネラルレジスタ B1(TRDGRB1)と同じ値にしてください。

⑪タイマ RD スタートレジスタ(TRDSTR:Timer RD start register)の設定

TRD0 をカウントさせるか、停止させるか設定します。

設定 bit	上:ビット名 下:シンボル	内容	今回の内容
bit7～4		″0000″を設定	0000
bit3	TRD1 カウント動作選択ビット csel1_trdstr	0:TRDGRA1 レジスタとのコンペア一致でカウント停止 1:TRDGRA1 レジスタとのコンペア一致後もカウント継続 今回は″1″を設定します。	1
bit2	TRD0 カウント動作選択ビット csel0_trdstr	0:TRDGRA0 レジスタとのコンペア一致でカウント停止 1:TRDGRA0 レジスタとのコンペア一致後もカウント継続 今回は″1″を設定します。	1
bit1	TRD1 カウント開始フラグ(注 4) tstart1_trdstr	0:カウント停止(注 2) 1:カウント開始 設定した瞬間から、TRD1 のカウントが開始されます。リセット同期 PWM モードでは TRD1 は使いませんので″0″を設定します。	0
bit0	TRD0 カウント開始フラグ(注 3) tstart0_trdstr	0:カウント停止(注 1) 1:カウント開始 設定した瞬間から、TRD0 のカウントが開始されます。″1″を設定します。	1

注 1. bit2 が“1”に設定されているとき、bit0 へ“0”を書いてください。
注 2. bit3 が“1”に設定されているとき、bit1 へ“0”を書いてください。
注 3. bit2 が“0”でコンペア一致信号(TRDIOA0)が発生したとき、“0”(カウント停止)になります。
注 4. bit3 が“0”でコンペア一致信号(TRDIOA1)が発生したとき、“0”(カウント停止)になります。

タイマ RD スタートレジスタ(TRDSTR)の設定値を下記に示します。

bit	7	6	5	4	3	2	1	0
設定値	0	0	0	0	1	1	0	**1**
16 進数	**0**				**d**			

23.6.2 motor関数

左モータ、右モータへ PWM を出力する関数です。

```
157 :  void motor( int data1, int data2 )
158 :  {
159 :      int    motor_r, motor_l, sw_data;
160 :
161 :      sw_data = dipsw_get() + 5;
162 :      motor_l = data1 * sw_data / 20;
163 :      motor_r = data2 * sw_data / 20;
164 :
165 :      /* 左モータ制御 */
166 :      if( motor_l >= 0 ) {
167 :          p2 &= 0xfd;
168 :          p2 |= 0x40;
169 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
170 :      } else {
171 :          p2 |= 0x02;
172 :          p2 &= 0xbf;
173 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
174 :      }
175 :
176 :      /* 右モータ制御 */
177 :      if( motor_r >= 0 ) {
178 :          p2 &= 0xf7;
179 :          p2 |= 0x80;
180 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
181 :      } else {
182 :          p2 |= 0x08;
183 :          p2 &= 0x7f;
184 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
185 :      }
186 :  }
```

(1) motor関数の使い方

motor 関数の使い方を下記に示します。

```
motor( 左モータの PWM 値 , 右モータの PWM 値 );
```

引数は、左モータの PWM 値と右モータの PWM 値をカンマで区切って入れます。値とモータの回転の関係を下記に示します。

値	説明
-100～-1	逆転します。-100 で 100%逆転です。-100 以上の値は設定できません。また、整数のみの設定になります。
0	モータが停止します。
1～100	正転します。100 で 100%正転です。100 以上の値は設定できません。また、整数のみの設定になります。

実際にモータに出力される割合を、下記に示します。

$$\text{左モータに出力される PWM} = \text{motor 関数で設定した左モータの PWM 値} \times \frac{\text{ディップスイッチの値+5}}{20}$$

$$\text{右モータに出力される PWM} = \text{motor 関数で設定した右モータの PWM 値} \times \frac{\text{ディップスイッチの値+5}}{20}$$

例えば、motor 関数で左モータに 80 を設定した場合、正転で 80%の回転をするかというと実はそうではありません。マイコンボード上にあるディップスイッチの値により、実際にモータへ出力される PWM の割合が変化します。
ディップスイッチが、”1100”(10 進数で 12)のとき、下記プログラムを実行したとします。

```
motor( -70 , 100 );
```

実際のモータに出力される PWM 値は、下記のようになります。

左モータに出力される PWM ＝ −70×(12＋5)÷20 ＝ −70×0.85 ＝ −59.5 ＝ −59%
右モータに出力される PWM ＝ 100×(12＋5)÷20 ＝ 100×0.85 ＝ 85%

左モータの計算結果は-59.5%ですが、小数点は計算できないので切り捨てられ整数になります。よって左モータに出力される PWM 値は逆転 59%、右モータに出力される PWM 値は正転 85%となります。

これから、上記の内容がどのように実行されるのか説明します。

(2) ディップスイッチの割合に応じて、PWM値を変更

```
161 :     sw_data = dipsw_get() + 5;        dipsw_get() = ディップスイッチの値0～15
162 :     motor_l = data1 * sw_data / 20;
163 :     motor_r = data2 * sw_data / 20;
```

161 行	sw_data 変数にディップスイッチの値＋5の値を代入します。ディップスイッチの値は 0～15 なので、sw_data 変数の値は、5～20 になります。
162 行	motor_l 変数は、左モータに加える PWM 値の割合を代入する変数です。data1 が motor 関数に設定した左モータの PWM 値です。 よって、次の計算を行って motor_l 変数に、左モータに加える PWM 値を設定します。 motor_l＝data1(motor 関数で設定した左モータの PWM)×sw_data／20 motor_l 変数の範囲は、-100～100 の値です。
163 行	motor_r 変数は、右モータに加える PWM 値の割合を代入する変数です。data2 が motor 関数に設定した右モータの PWM 値です。 よって、次の計算を行って motor_r 変数に、右モータに加える PWM 値を設定します。 motor_r＝data2(motor 関数で設定した右モータの PWM)×sw_data／20 motor_r 変数の範囲は、-100～100 の値です。

(3) 左モータ制御

左モータを制御する部分です。左モータの PWM は P2_2 端子から出力します。P2_2 端子から出力する PWM の設定は、タイマ RD ジェネラルレジスタ B0(TRDGRB0)に PWM 値を設定します。ただし、直接設定するのではなく、バッファレジスタであるタイマ RD ジェネラルレジスタ D0(TRDGRD0)に設定します。

```
165 :      /* 左モータ制御 */
166 :      if( motor_l >= 0 ) {
167 :          p2 &= 0xfd;              p2 = p2 AND 0xfdという意味です
168 :          p2 |= 0x40;              p2 = p2 OR  0x40という意味です
169 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
170 :      } else {
171 :          p2 |= 0x02;              p2 = p2 OR  0x02という意味です
172 :          p2 &= 0xbf;              p2 = p2 AND 0xbfという意味です
173 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
174 :      }
```

166 行	左モータの PWM 値の割合が正の数か負の数かをチェックします。 正の数なら 167～169 行、負の数なら 171～173 行を実行します。
167 行 ～ 169 行	正の数なら 167～169 行を実行します。 P2_1="0"、P2_6="1"を設定し、P2_2 端子から PWM 出力すると、PWM の割合に応じてモータが正転します。 167 行目で下表の計算を行い、P2_1 端子を"0"にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
AND 値	1	1	1	1	1	1	**0**	1
結果	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	**0**	P2_0

168 行目で下表の計算を行い、P2_6 端子を"1"にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
OR 値	0	**1**	0	0	0	0	0	0
結果	P2_7	**1**	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0

169 行目で次の計算を行って、タイマ RD ジェネラルレジスタ D0(TRDGRD0)へ PWM 値を設定しています。小数点が出た場合は、切り捨てられます。

$$TRDGRD0 = (PWM\_CYCLE - 1) \times \frac{motor\_l\ (0 \sim 100)}{100}$$

$$= 39998 \times \frac{motor\_l\ (0 \sim 100)}{100}$$

例えば motor_l=80 なら、TRDGRD0 は次のように計算されます。

TRDGRD0＝39998×80／100＝31998.4＝31998

171 行
～
173 行

負の数なら 171～173 行を実行します。
P2_1=″1″、P2_6=″0″を設定し、P2_2 端子から PWM 出力すると、PWM の割合に応じてモータが逆転します。
171 行目で下表の計算を行い、P2_1 端子を″1″にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
OR 値	0	0	0	0	0	0	**1**	0
結果	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	**1**	P2_0

172 行目で下表の計算を行い、P2_6 端子を″0″にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
AND 値	1	**0**	1	1	1	1	1	1
結果	P2_7	**0**	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0

173 行目で次の計算を行って、タイマ RD ジェネラルレジスタ D0(TRDGRD0)へ PWM 値を設定しています。小数点が出た場合は、切り捨てられます。

$$TRDGRD0 = (PWM\_CYCLE - 1) \times \frac{-motor\_l\ (-1 \sim -100)}{100}$$

$$= 39998 \times \frac{-motor\_l\ (-1 \sim -100)}{100}$$

ポイントは、motor_l 変数が負の数ということです。回路的には P2_1=″1″、P2_6=″0″で逆転の設定になっています。そのため motor_l は、正の数に直して計算します。直し方は、計算式の中で「-motor_l」としています。例えば motor_l=-50 なら、TRDGRD0 は次のように計算されます。

TRDGRD0＝39998×{-(-50)}／100＝39998×50／100＝19999

(4) 右モータ制御

右モータを制御する部分です。右モータの PWM は P2_4 端子から出力します。P2_4 端子から出力する PWM の設定は、タイマ RD ジェネラルレジスタ A1(TRDGRA1)に PWM 値を設定します。ただし、直接設定するのではなく、バッファレジスタであるタイマ RD ジェネラルレジスタ C1(TRDGRC1)に設定します。

```
176 :      /* 右モータ制御 */
177 :      if( motor_r >= 0 ) {
178 :          p2 &= 0xf7;              p2 = p2 AND 0xf7という意味です
179 :          p2 |= 0x80;              p2 = p2 OR  0x80という意味です
180 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
181 :      } else {
182 :          p2 |= 0x08;              p2 = p2 OR  0x08という意味です
183 :          p2 &= 0x7f;              p2 = p2 AND 0x7fという意味です
184 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
185 :      }
```

177 行	右モータの PWM 値の割合が正の数か負の数かをチェックします。 正の数なら 178～180 行、負の数なら 182～184 行を実行します。
178 行 ～ 180 行	正の数なら 178～180 行を実行します。 P2_3="0"、P2_7="1"を設定し、P2_4 端子から PWM 出力すると、PWM の割合に応じてモータが正転します。 178 行目で下表の計算を行い、P2_3 端子を"0"にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
AND 値	1	1	1	1	**0**	1	1	1
結果	P2_7	P2_6	P2_5	P2_4	**0**	P2_2	P2_1	P2_0

179 行目で下表の計算を行い、P2_7 端子を"1"にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
OR 値	**1**	0	0	0	0	0	0	0
結果	**1**	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0

180 行目で次の計算を行って、タイマ RD ジェネラルレジスタ C1(TRDGRC1)へ PWM 値を設定しています。小数点が出た場合は、切り捨てられます。

$$TRDGRC1 = (PWM\_CYCLE - 1) \times \frac{motor\_r\ (0 \sim 100)}{100}$$

$$= 39998 \times \frac{motor\_r\ (0 \sim 100)}{100}$$

例えば例えば motor_r=20 なら、TRDGRC1 は次のように計算されます。

TRDGRC1＝39998×20／100＝7999.6＝7999

182 行
～
184 行

負の数なら 182～184 行を実行します。
P2_3="1"、P2_7="0"を設定し、P2_4 端子から PWM 出力すると、PWM の割合に応じてモータが逆転します。
182 行目で下表の計算を行い、P2_3 端子を"1"にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
OR 値	0	0	0	0	**1**	0	0	0
結果	P2_7	P2_6	P2_5	P2_4	**1**	P2_2	P2_1	P2_0

183 行目で下表の計算を行い、P2_7 端子を"0"にします。

bit	7	6	5	4	3	2	1	0
元の値 (ポート 2)	P2_7	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0
AND 値	**0**	1	1	1	1	1	1	1
結果	**0**	P2_6	P2_5	P2_4	P2_3	P2_2	P2_1	P2_0

184 行目で次の計算を行って、タイマ RD ジェネラルレジスタ C1(TRDGRC1)へ PWM 値を設定しています。小数点が出た場合は、切り捨てられます。

$$TRDGRC1 = (PWM\_CYCLE - 1) \times \frac{-motor\_r\ (-1 \sim -100)}{100}$$

$$= 39998 \times \frac{-motor\_r\ (-1 \sim -100)}{100}$$

ポイントは、motor_r 変数が負の数だということです。回路的には P2_3="1"、P2_7="0"で逆転の設定になっています。そのため motor_r は、正の数に直して計算します。直し方は、計算式の中で「-motor_r」としています。例えば motor_r=-90 なら、TRDGRC1 は次のように計算されます。

TRDGRC1＝39998×{-(-90)}／100＝39998×90／100＝35998.2＝35998

23.6.3 main関数

```
 43 :  void main( void )
 44 :  {
 45 :      init();                              /* 初期化                       */
 46 :      asm(" fset I ");                     /* 全体の割り込み許可           */
 47 :
 48 :      while( 1 ) {
 49 :          motor( 100 , 0);
 50 :          timer( 1000 );
 51 :          motor( 0, 80 );
 52 :          timer( 1000 );
 53 :          motor( -60, 0 );
 54 :          timer( 1000 );
 55 :          motor( 0, -40 );
 56 :          timer( 1000 );
 57 :          motor( 0, 0 );
 58 :          timer( 1000 );
 59 :      }
 60 :  }
```

45 行	init 関数を実行します。 init 関数では、ポートの入出力設定、タイマ RB による 1ms ごとの割り込み設定、タイマ RD によるリセット同期 PWM モードの設定を行っています。
46 行	全体の割り込みを許可する命令です。 今回は、タイマ RB による 1ms ごとの割り込みを許可するために実行しています。
48 行	while 文で無限ループを作っています。カッコでくくられた 49 行～58 行が実行され続けます。
49 行	motor 関数を使って、左モータ 100%、右モータ 0%で回転させます。ただし、実際のモータに出力される PWM 波形は、ディップスイッチの割合も加わります。
50 行	1000ms の時間、この行で待ちます。
51 行～ 58 行	同様にモータを制御します。