デュアルショック 2
制御基板製作・
プログラム解説マニュアル
(R8C/38A 版)

本マニュアルで 説明している内容	PlayStation 2 用のアナログコントローラ「デュアルショック 2」の状態を R8C/38A マイコンで読み込み、ロボットなどを制御する基板の製作、プログラムの解説をしています。
本基板の 対象マイコンボード	RY_R8C38 ボード
本基板の制作(結線)に ついての説明	本マニュアルで解説
本基板のプログラムに ついての説明	本マニュアルで解説

本マニュアルの内容は、独自の解析によるものでメーカの保証外となります。自己責任での取り組みとなりますので、ご了承ください。

デュアルショック(SCPH-1200)は動作しません。必ずデュアルショック 2(SCPH-10010)を使用してください。

第 1.02 版
2015.10.17
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

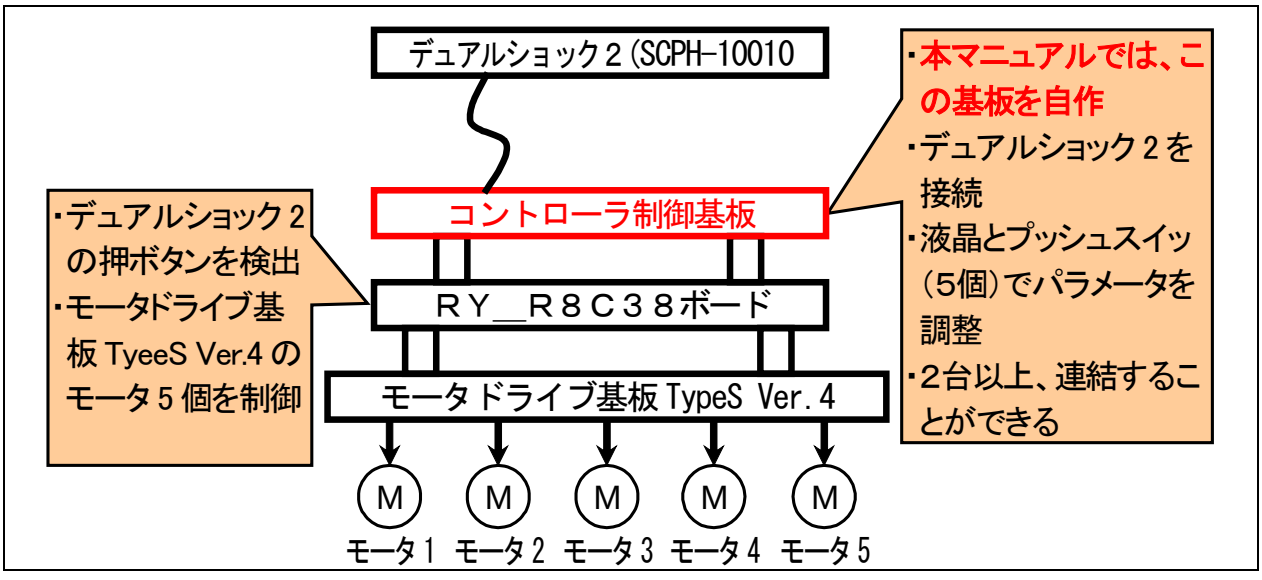
1.1 目的

本マニュアルは、PlayStation 用のデュアルショック 2(型式:SCPH-10010、以下コントローラ)のボタン操作を RY_R8C38 ボード(R8C/38A マイコン)で読み込み、モータなどを制御することを目的とした内容です。
※デュアルショック(SCPH-1200)は動作しません。必ずデュアルショック 2(SCPH-10010)を使用してください。

▲デュアルショック 2(SCPH-10010)

1.2 構成

本マニュアルは、下記構成で説明しています。**本マニュアルでは、コントローラ制御基板を自作します**。



1.3 資料の参照先

それぞれの基板、機器についての製作方法、内容は、マイコンカーラリー販売サイト(https://www2.himdx.net/mcr/)の下表のマニュアルを参照してください。

内容	マニュアル
プログラムの開発環境について	ルネサス統合開発環境　操作マニュアル(R8C/38A 版)
モータドライブ基板 TypeS Ver.4 について	●製作について モータドライブ基板 TypeS Ver.4　製作マニュアル ●プログラムについて モータドライブ基板 TypeS Ver.4 アナログセンサ基板 TypeS Ver.2 プログラム解説マニュアル
液晶、プッシュスイッチ、データフラッシュについて	●製作について 液晶・microSD 基板　製作マニュアル ●プログラムについて 液晶・microSD 基板　kit12_38a プログラム解説マニュアル　液晶編

2. コントローラ制御基板

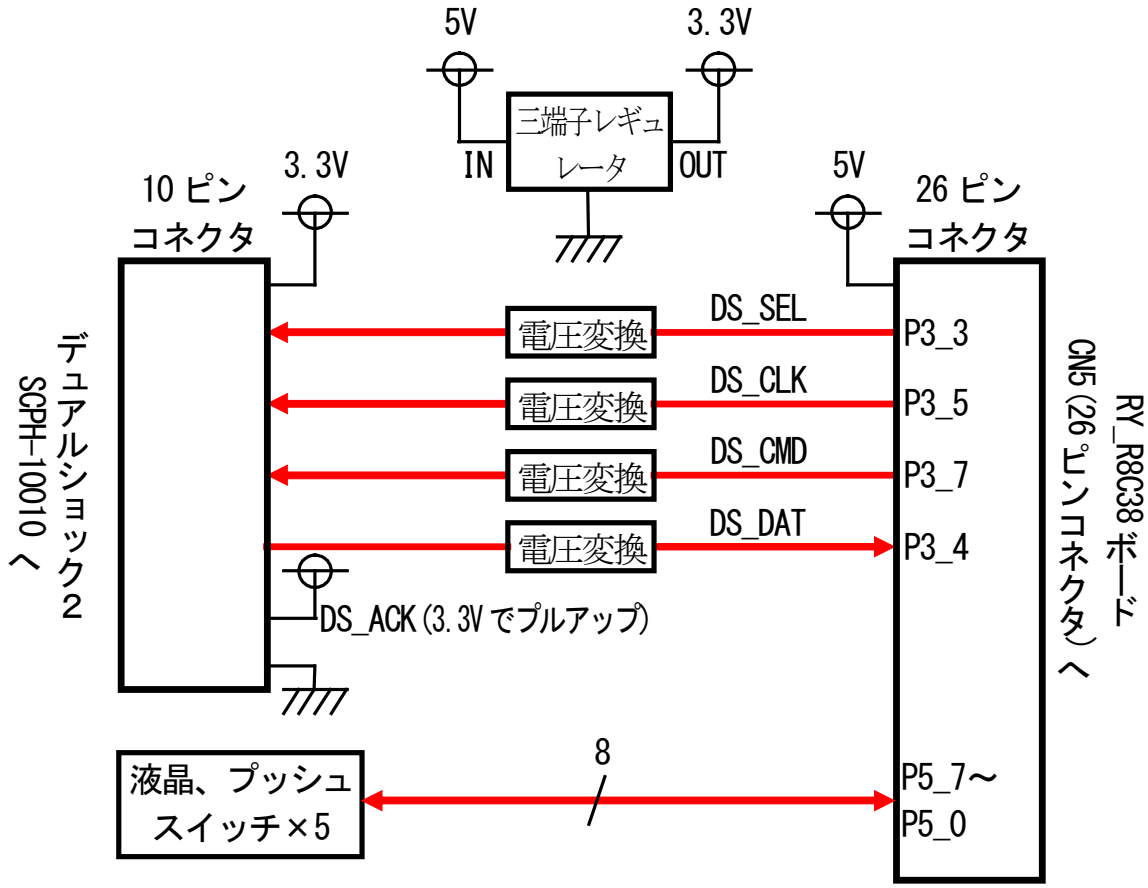
2.1 コントローラとの接続

コントローラの動作電圧は、3.6V です。そのため、三端子レギュレータで 3.3V を生成します。3.6V を出力する三端子レギュレータはほとんどありませんので、一般的な 3.3V 出力を使用します。差は 0.3V なので、ほとんど問題ありません。

R8C/38A マイコンの動作電圧は 5V、コントローラの動作電圧は 3.3V です。直接信号のやり取りはできません。そのため、次のような変換が必要です。

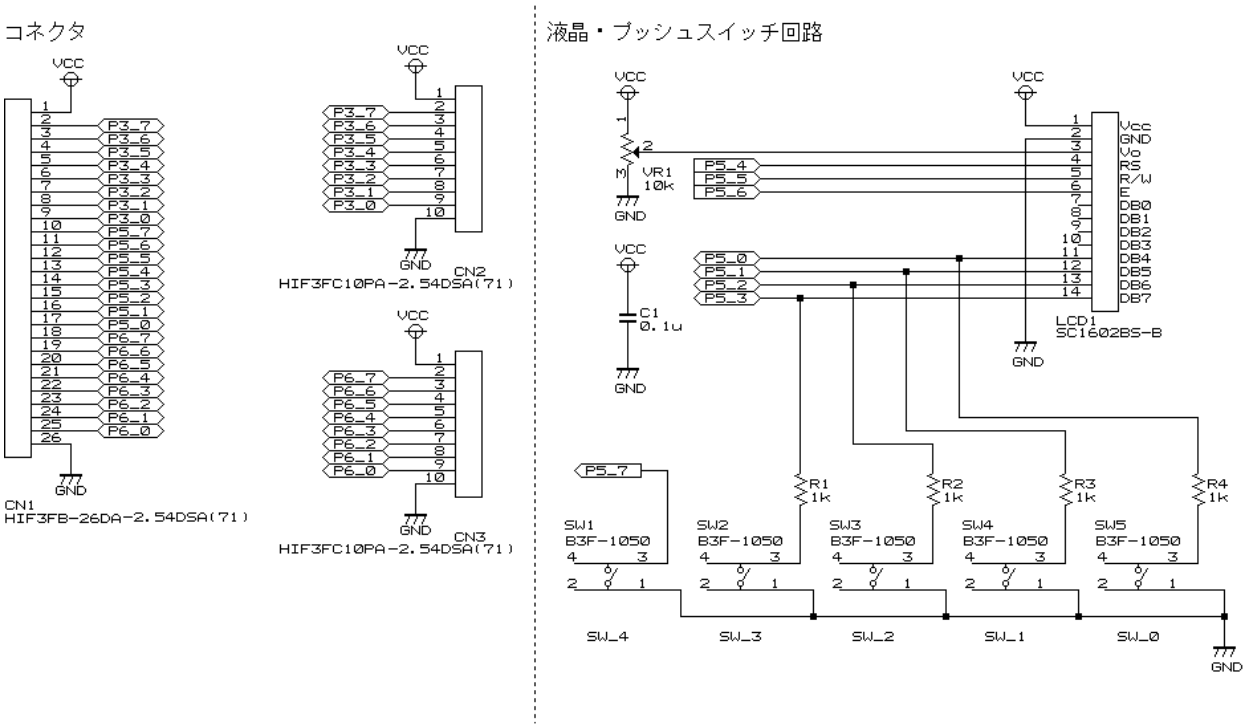
①R8C/38A マイコン→コントローラへの信号は、5V→3.3V の電圧変換
②コントローラ→R8C/38A マイコンへの信号は、3.3V→5V の電圧変換

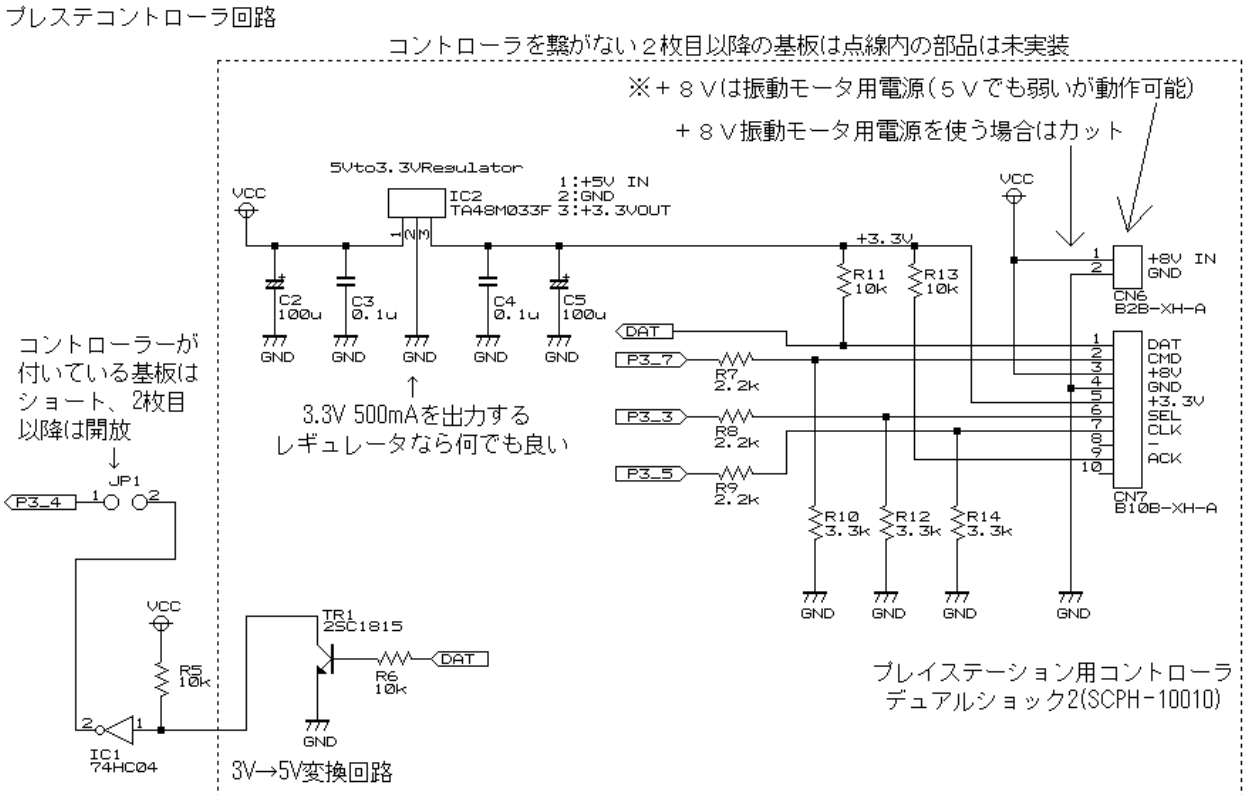
今回は、①は抵抗による分圧、②はトランジスタで電圧変換しています。



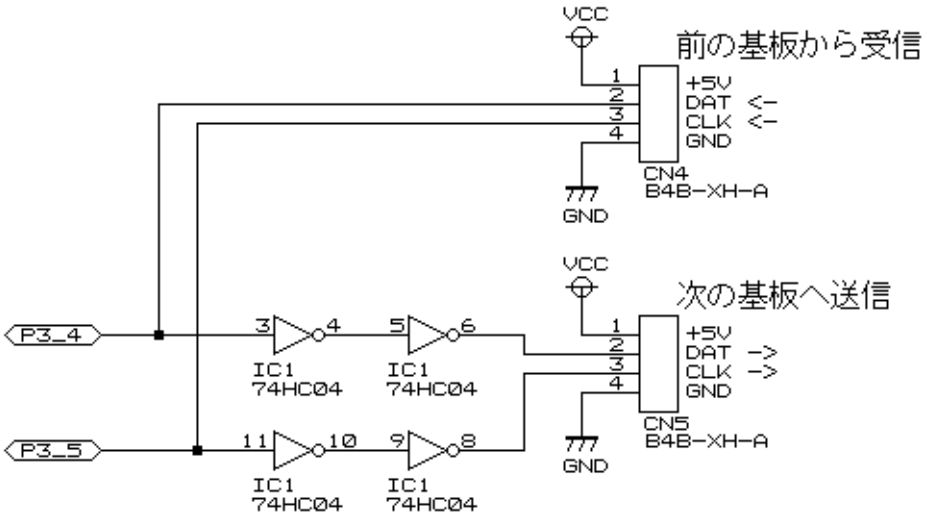
2.2 回路図

※液晶とプッシュスイッチ(5 個)の接続は、液晶・microSD 基板(Ver.2 含む)と同様です。

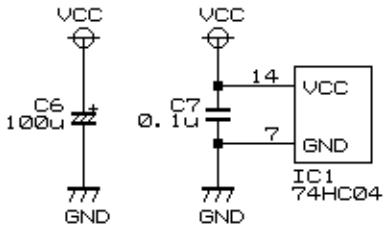


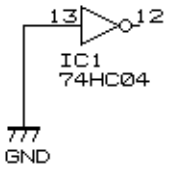


次基板出力回路



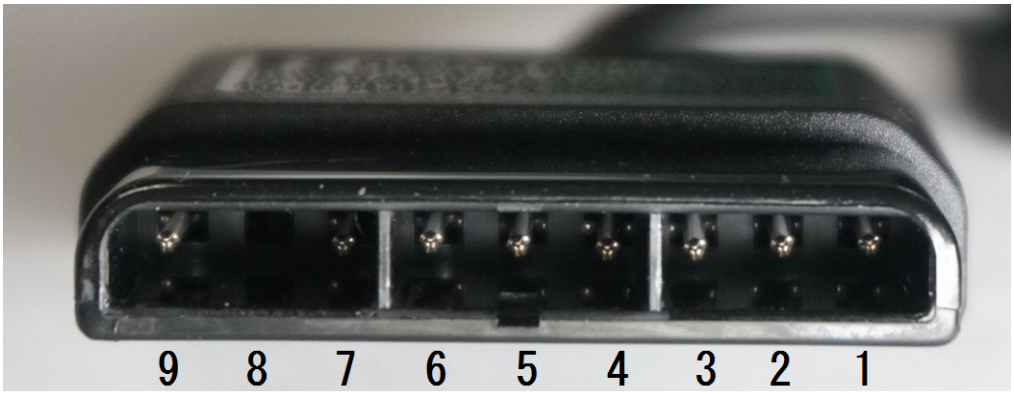
IC電源回路



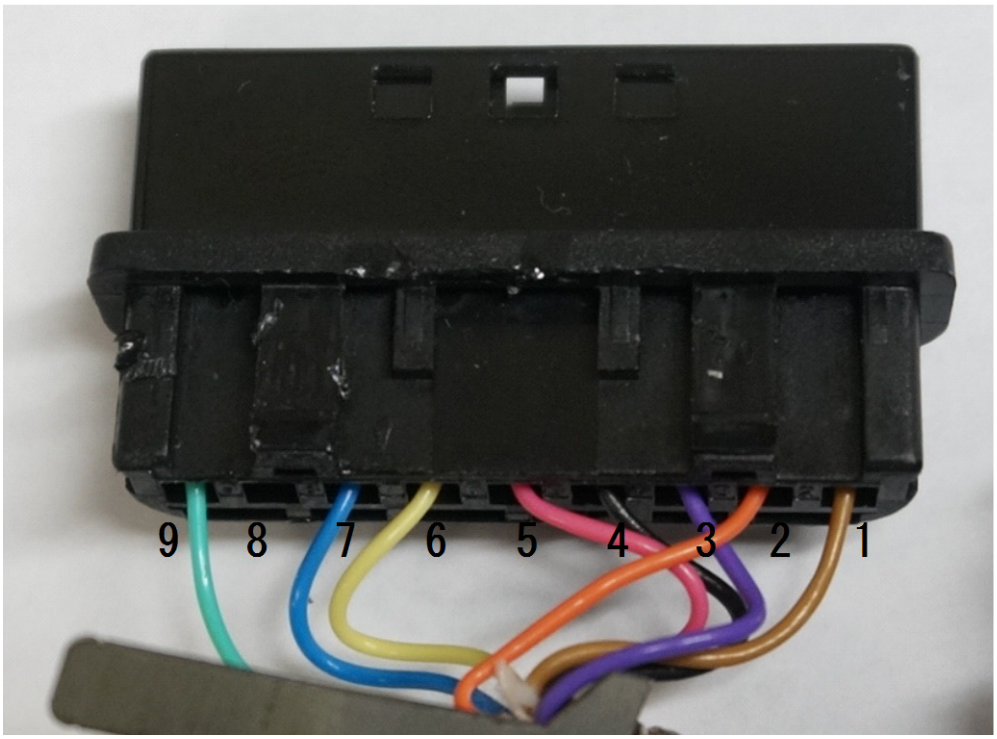


2.3 コントローラのコネクタ

コントローラ側のオスコネクタのピン配置を下記に示します。コネクタ下部の金属端子は、4 ピンとケーブルのシールドに接続されています。



コネクタカバーを開けたところを、下記に示します。

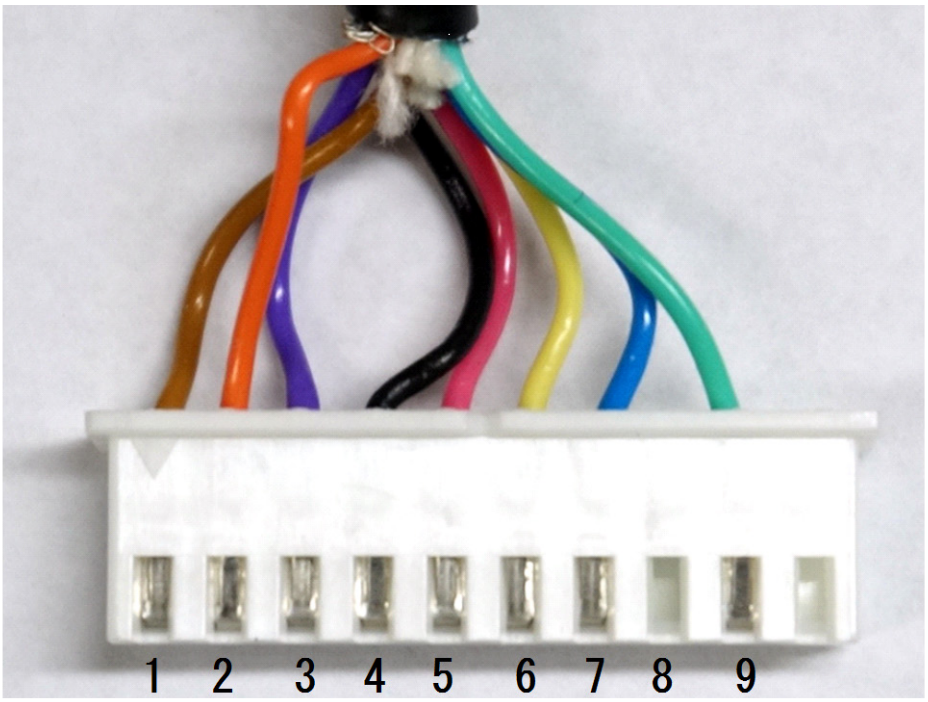


ピンの名称を下表に示します。

番号	線の色※	名称	方向	詳細
1	茶	DAT	ﾏｲｺﾝ←ｺﾝﾄﾛｰﾗ	データ
2	橙	CMD	ﾏｲｺﾝ→ｺﾝﾄﾛｰﾗ	コマンド
3	紫	7.6V	電源	振動モータ用の電源です。5V でも振動しますが弱くなります。
4	黒	GND		GND
5	赤	3.6V	電源	コントローラを動作させるための電源です。3.3V でも可能です。
6	黄	SEL	ﾏｲｺﾝ→ｺﾝﾄﾛｰﾗ	コントローラを選択する信号です。"0"で有効、"1"で無効です。
7	青	CLK	ﾏｲｺﾝ→ｺﾝﾄﾛｰﾗ	クロック信号です。コントローラは CLK の立ち上がりでコマンドを読み取ります。標準のクロックの間隔は 250μs ですが、遅くとも動作します。
8	無し	無し		
9	緑	ACK	ﾏｲｺﾝ←ｺﾝﾄﾛｰﾗ	アクノリッジ。コントローラがデータを読み込むと 2μs 以上"0"になります。今回は接続しません。

※色は、異なることがあります。現品のコネクタの番号と色を必ず確認してください。

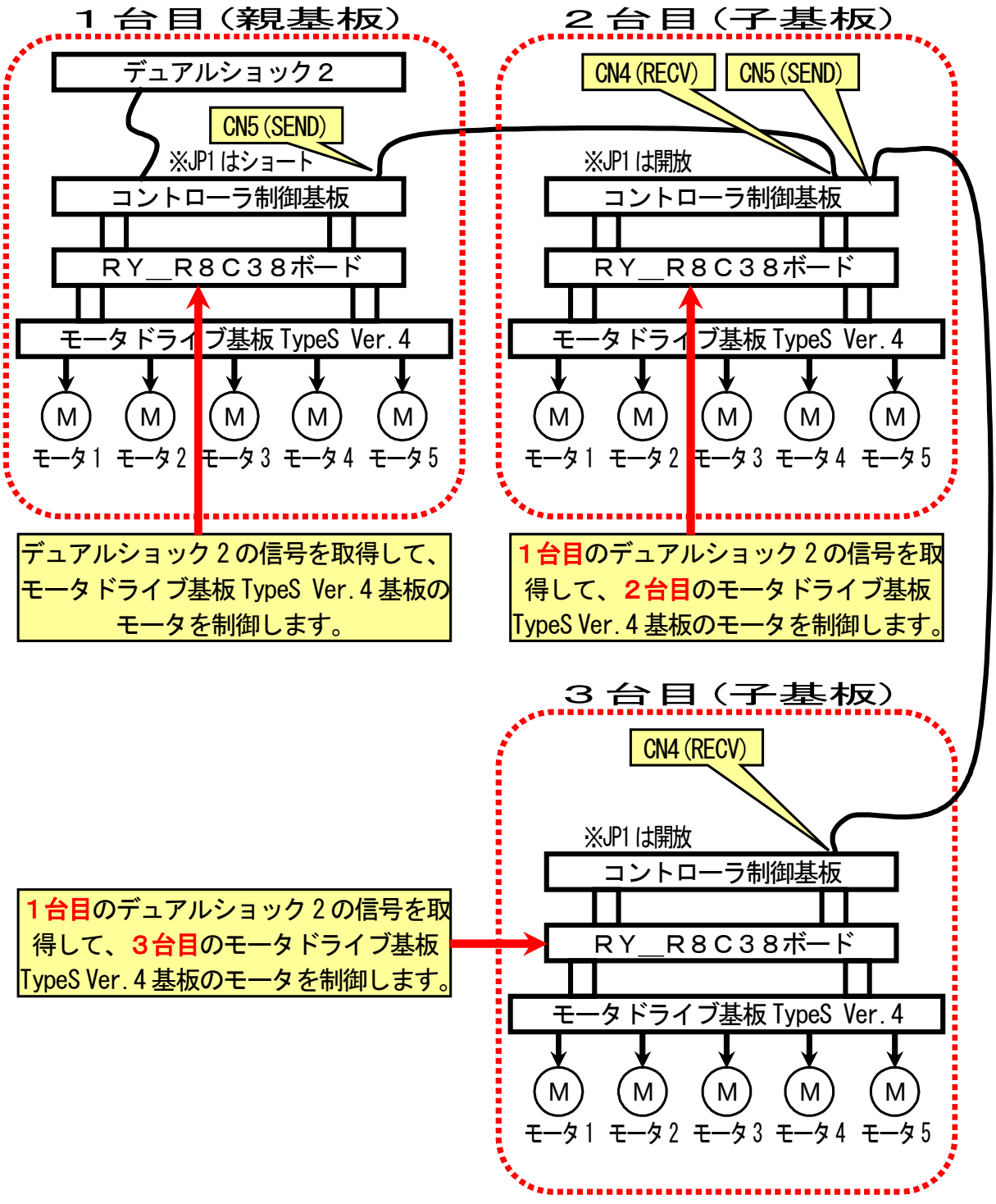
コントローラのコネクタを切断し、日本圧着端子製造(株)(JST)の XH コネクタ 10 ピンに取り付け直した例を、下記に示します。



2.4 制御基板を連結させる

モータドライブ基板 TypeS Ver.4 は、1 枚で 5 個のモータを制御することのできる基板ですが、ロボット競技大会ではモータを 6 個以上制御したいことがほとんどです。今回の基板は、1 台目の送信コネクタ(CN5)と、2 台目の受信コネクタ(CN4)をつなぎ合わせて、1つのコントローラで 1 台目と 2 台目のモータドライブ基板 TypeS Ver.4 を制御することができます。2 台目と 3 台目も同様に接続して増やすことができます。コントローラからの信号を受信しているだけなので、論理的には無限に連結させることができます。

3枚連結させるときの概要を下記に示します。なお、**コントローラが繋がっている 1 台目の基板を親基板、2台目以降を子基板と呼びます**。



3. プログラム「scph_10010.c」の解説

3.1 概要

「scph_10010.c」は、R8C/38A マイコンで PlayStation 用のデュアルショック 2(型式:SCPH-10010)と通信を行い、ボタンとジョイスティックの情報を読み取るプログラムです。また、コントローラに付いている振動モータを制御することができます。

本プログラムは単体では使用できません。各自が作成したメインプログラムから本プログラムにある関数を呼び出し使用します(今回は「ps_controller_types4_38a.c」から、各関数を呼び出します)。

コントローラ処理は、R8C/38A マイコンの「チップセレクト付クロック同期形シリアル I/O (SSU)のクロック同期通信モード」を使用します。SSU は、コントローラ処理以外で使用することはできません。

コントローラとの通信処理は、0.25msごとに行います。今回はタイマ RB を使って 0.25mごとに割り込みを発生させ、割り込みプログラム内で実行します。0.25ms ごとに実行できれば、タイマ RB を使わなくても構いません。

3.2 関数

本プログラムは、次の関数を用意しています。

■initDS 関数

書式	void initDS(int mode);
内容	コントローラや変数などを初期化します。割り込みを許可したあとに実行します。
引数	0　　:コントローラを接続している1台目の基板 (0 以外):2 台目以降(コントローラは接続していない)
戻り値	なし
例	initDS(0);

■setVaibrateData 関数

書式	void setVaibrateData(int mini,int big);
内容	振動モータの振動を制御します。
引数	小モータの振動[0:OFF 1:ON]，大モータの振動[0～255:0～100%で振動]
戻り値	なし
例	setVaibrateData(1, 128);　小は ON、大は 50%で振動

■dsProcess 関数

書式	void dsProcess(void);
内容	コントローラの制御を行います。割り込みなどで 0.25ms ごとに実行します。
引数	なし
戻り値	なし
例	dsProcess();　　// 0.25ms ごとに実行

■getDSKeyData 関数

<table>
<tr><td>書式</td><td>unsigned int getDSKeyData(void);</td></tr>
<tr><td>内容</td><td>コントローラのキーデータを取得します。ボタンは 16 個あります。</td></tr>
<tr><td>引数</td><td>なし</td></tr>
<tr><td>戻り値</td><td>unsigned int キーデータ(16bit)
キーデータの 1bit が一つのボタンに対応しています。

<table>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td>←</td><td>↓</td><td>→</td><td>↑</td><td>START</td><td>右ジョイスティックの押しボタン</td><td>左ジョイスティックの押しボタン</td><td>SELECT</td></tr>
</table>

<table>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>□</td><td>×</td><td>○</td><td>△</td><td>R1</td><td>L1</td><td>R2</td><td>L2</td></tr>
</table>
</td></tr>
<tr><td>例</td><td>

```
d = getDSKeyData();
if( d & DS_KEY_LEFT ) {
   // 左キー押されたらここを実行
}
```

</td></tr>
<tr><td>キー定義</td><td>「scph_10010.h」に下記内容を定義しています。例の太字部分を、下記の太字部分に書き換えて、必要なボタンに置き換えてください。


```
#define DS_KEY_LEFT         (0x01 << 15)
#define DS_KEY_DOWN         (0x01 << 14)
#define DS_KEY_RIGHT        (0x01 << 13)
#define DS_KEY_UP           (0x01 << 12)
#define DS_KEY_START        (0x01 << 11)
#define DS_KEY_R_JOYSTICK   (0x01 << 10)
#define DS_KEY_L_JOYSTICK   (0x01 <<  9)
#define DS_KEY_SELECT       (0x01 <<  8)
#define DS_KEY_SHIKAKU      (0x01 <<  7)
#define DS_KEY_BATSU        (0x01 <<  6)
#define DS_KEY_MARU         (0x01 <<  5)
#define DS_KEY_SANKAKU      (0x01 <<  4)
#define DS_KEY_R1           (0x01 <<  3)
#define DS_KEY_L1           (0x01 <<  2)
#define DS_KEY_R2           (0x01 <<  1)
#define DS_KEY_L2           (0x01 <<  0)
```

</td></tr>
</table>

■getDSJoyStickData 関数

書式	int getDSJoyStickData(int index);
内容	コントローラのジョイスティックの位置を取得します。 ジョイスティックは、左側の上下の値、左右の値、右側の上下の値、左右の値の 4 種類あります。
引数	int ジョイスティックの位置 どのジョイスティックの値を取得するか指定します。 DS_LEFT_X ： 左側の左右(X)の値を取得 DS_LEFT_Y ： 左側の上下(Y)の値を取得 DS_RIGHT_X ： 右側の左右(X)の値を取得 DS_RIGHT_Y ： 右側の上下(Y)の値を取得
戻り値	値 int -100～100 戻り値には-100～100 の値が返ってきます。 X の場合、左が-100、中心が 0、右が 100 となります。 Y の場合、下が-100、中心が 0、上が 100 となります。

例	i = getDSJoyStickData(DS_LEFT_X); // iには、左側ジョイスティックのXの値が入る(-100～100)
	
	if(getDSJoyStickData(DS_RIGHT_Y) <= -50) {
	// 右側ｼﾞｮｲｽﾃｨｯｸの Y が-50 以下ならここを実行
	}

3.3 端子を変えるときは

基板を自作するとき、コントローラの SEL 端子と接続している P3_3 端子は、変更することができます。端子を変更するときは、「scph_10010.h」の下記部分を変更します。

```
80 : #define DS_SEL_PORT      p3_3      /* SEL 端子のポートレジスタ      */
81 : #define DS_SEL_PORTDIR   pd3_3     /* SEL 端子のポート方向ﾚｼﾞｽﾀ     */
```

3.4 コントローラが不安定なときは

コントローラを使用していて、キーデータがうまく取れないことがある、振動しないことがあるなど、コントローラが不安定なときは、「scph_10010.c」の次の部分を変更してみてください。

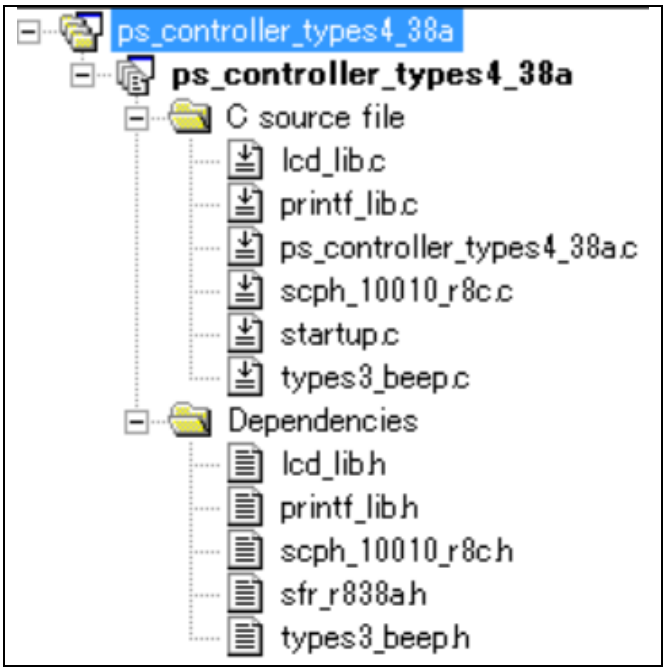
```
84 : #define DS_SEND_WAIT     32        /* 1=0.25ms 64 が標準            */
```

84 行の、「DS_SEND_WAIT」は、コントローラとの通信間隔です。数字 1 あたり 0.25ms です。よって、「32×0.25＝8ms」間隔で通信をします。

規格は 16ms ごとに通信すると決められていますが、今回は 8ms ごとにして、コントローラで操作した内容ができるだけ早くマイコンで読み込めるようにしています。ただし、規格より小さい値なので、コントローラによってはうまくキー情報が読み込めないなど不安定な状態になるかもしれません。このときは、数字を大きくしてください(+10 ずつくらい)。コントローラとの通信間隔は遅くなりますが安定します。64 以上にしても不安定な場合は、コントローラ以外の原因が考えられます。その他の部分を確認してください。

4. ワークスペース「ps_controller_types4_38a」

4.1 プロジェクトの構成



1	lcd_lib.c lcd_libhc	液晶制御ライブラリです。液晶を使用する場合は、このファイルを追加します。 ファイルの位置→C:¥WorkSpace¥common_r8c38a¥lcd_lib.c
2	printf_lib.c	通信をするための設定、printf 関数の出力先、scanf 関数の入力元を通信にするための設定を行っています。 **今回は printf 文や scanf 文は使いませんが、液晶の文字表示でこのファイルを追加しないとエラーが出るため、追加しておきます。** ファイルの位置→C:¥WorkSpace¥common_r8c38a¥printf_lib.c
3	ps_controller_ types4_38a.c	実際に制御するプログラムが書かれています。R8C/38A マイコンの内蔵周辺機能(SFR)の初期化も行います。 ファイルの位置→C:¥WorkSpace¥ps_controller_types4_38a¥ ps_controller_types4_38a¥ps_controller_types4_38a.c
4	scph_10010_r8c.c scph_10010_r8c.h	デュアルショック 2(SCPH-10010)を制御するためのライブラリです。デュアルショック 2 を使用する場合は、このファイルを追加します。 ファイルの位置→C:¥WorkSpace¥ps_controller_types4_38a¥ ps_controller_types4_38a¥scph_10010_r8c.c
5	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。 ファイルの位置→C:¥WorkSpace¥ps_controller_types4_38a¥ ps_controller_types4_38a¥startup.c
6	types3_beep.c types3_beep.h	モータドライブ基板 TypeS Ver.4 に搭載しているブザーを制御するプログラムです。 ファイルの位置→C:¥WorkSpace¥ps_controller_types4_38a¥ ps_controller_types4_38a¥types3_beep.c
7	sfr_r838a.h	R8C/38A 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 Register)を定義したファイルです。 ファイルの位置→C:¥WorkSpace¥common_r8c38a¥sfr_r838a.h

4.2 プログラムリスト「ps_controller_types4_38a.c」

```
/************************************************************************/
/* 対象マイコン R8C/38A                                                 */
/* ﾌｧｲﾙ内容     モータドライブ基板TypeS Ver.4+デュアルショック2          */
/*              を使った制御プログラム                                  */
/* バージョン   Ver.1.00                                                */
/* Date         2014.03.21                                              */
/* Copyright    ルネサスマイコンカーラリー事務局                        */
/************************************************************************/

/*
本プログラムは、
●モータドライブ基板TypeS Ver.4
●デュアルショック2　制御基板
を使用したロボットを動作させるプログラムです。
*/

/*====================================*/
/* インクルード                       */
/*====================================*/
#include <stdio.h>
#include "sfr_r838a.h"                  /* R8C/38A SFRの定義ファイル    */
#include "types3_beep.h"                /* ブザー追加                   */
#include "lcd_lib.h"                    /* 液晶表示用追加               */
#include "printf_lib.h"                 /* printf使用ライブラリ         */
#include "scph_10010_r8c.h"             /* SCPH-10010処理               */

/*====================================*/
/* シンボル定義                       */
/*====================================*/
/* 定数設定 */
#define TRC_MOTOR_CYCLE     20000       /* 左前,右前モータPWMの周期     */
                                        /* 50[ns] * 20000 = 1.00[ms]    */
#define TRD_MOTOR_CYCLE     20000       /* 左後,右後,ｻｰﾎﾞﾓｰﾀPWMの周期   */
                                        /* 50[ns] * 20000 = 1.00[ms]    */
#define FREE                1           /* モータモード　フリー         */
#define BRAKE               0           /* モータモード　ブレーキ       */

/*====================================*/
/* プロトタイプ宣言                   */
/*====================================*/
void init( void );
unsigned char dipsw_get( void );
unsigned char dipsw_get2( void );
unsigned char pushsw_get( void );
unsigned char cn6_get( void );
void led_out( unsigned char led );
void motor2_r( int accele_l, int accele_r );
void motor2_f( int accele_l, int accele_r );
void motor_mode_r( int mode_l, int mode_r );
void motor_mode_f( int mode_l, int mode_r );
void servoPwmOut( int pwm );

/*====================================*/
/* グローバル変数の宣言               */
/*====================================*/
int             pattern;                /* マイコンカー動作パターン     */
unsigned long   cnt1;                   /* タイマ用                     */
unsigned long   cnt_lcd;                /* 液晶表示タイミング用         */

/* エンコーダ関連 */
long            lEncoderTotal;          /* 積算値保存用                 */
int             iEncoder;               /* 10ms毎の最新値               */
unsigned int    uEncoderBuff;           /* 計算用　割り込み内で使用     */

/* TRCレジスタのバッファ */
unsigned int    trcgrb_buff;            /* TRCGRBのバッファ             */
unsigned int    trcgrd_buff;            /* TRCGRDのバッファ             */

/* モータドライブ基板TypeS Ver.4上のLED、ディップスイッチ制御 */
unsigned char   types_led;              /* LED値設定                    */
unsigned char   types_dipsw;            /* ディップスイッチ値保存       */

/* モータのPWM */
int             m1_pwm;                 /* モータ1(CN10)のPWM           */
int             m2_pwm;                 /* モータ2(CN11)のPWM           */
int             m3_pwm;                 /* モータ3(CN12)のPWM           */
int             m4_pwm;                 /* モータ4(CN13)のPWM           */
int             m5_pwm;                 /* モータ5(CN14)のPWM           */
```

モータを制御する PWM 周期を設定します。値は、「50ns×設定値」になります。今回は 20000 を設定しているので 1.00ms が PWM 周期になります。通常は 1ms で問題ありません。また回路の応答速度の限界で、0.5ms 以下にすると、モータが制御できなくなることがあります。

ロボットでは、モータを 1 個 1 個制御することが多いので、今回は motor 関数を使わなくても、良いようにしました。
モータの PWM を設定するときは、m1_pwm～m5_pwm 変数に PWM 値をセットしてください。値は-100～100 で、プラスが正転、マイナスが逆転、0 で停止になります。

```
 80 :  /*****************************************************************************/
 81 :  /* メインプログラム                                                          */
 82 :  /*****************************************************************************/
 83 :  void main( void )
 84 :  {
 85 :      int i, ds_mini, ds_big;
 86 :      unsigned int d;
 87 :
 88 :      /* マイコン機能の初期化 */
 89 :      init();                             /* 初期化                           */
 90 :      asm(" fset I ");                    /* 全体の割り込み許可               */
 91 :      initBeepS();                        /* ブザー関連処理                   */
 92 :      initLcd();                          /* 液晶初期化                       */
 93 :      initDS( 0 );                        /* 0:メイン 1:2台目以降             */
 94 :
 95 :      /* ロボットの状態初期化 */
 96 :      motor_mode_f( BRAKE, BRAKE );
 97 :      motor_mode_r( BRAKE, BRAKE );
 98 :      motor2_f( 0, 0 );
 99 :      motor2_r( 0, 0 );
100 :      servoPwmOut( 0 );
101 :      setBeepPatternS( 0x8000 );
102 :
103 :      while( 1 ) {
104 :
105 :      /* 液晶表示処理 */
106 :      if( cnt_lcd >= 100 ) {    // 液晶表示は100msごと
107 :          cnt_lcd = 0;
108 :
109 :          /* LCDにコントローラのボタンとジョイスティック(JS)の値 表示 */
110 :          lcdPosition( 0, 0 );
111 :                   /* 0123456789abcdef 1行16文字 */
112 :          lcdPrintf( "%04x %+4d %+4d  ",
113 :              getDSKeyData(),                 /* ボタン押下データ表示 */
114 :              getDSJoyStickData(DS_LEFT_X),   /* JS左の横方向の値     */
115 :              getDSJoyStickData(DS_LEFT_Y) );     /* JS左の縦方向の値     */
116 :          lcdPosition( 0, 1 );
117 :                   /* 0123456789abcdef 1行16文字 */
118 :          lcdPrintf( "     %+4d %+4d  ",
119 :              getDSJoyStickData(DS_RIGHT_X),  /* JS右の横方向の値     */
120 :              getDSJoyStickData(DS_RIGHT_Y) );    /* JS右の縦方向の値     */
121 :      }
122 :
123 :      // デュアルショック操作処理
124 :
125 :      d = getDSKeyData();                     /* キー情報の取得           */
126 :
127 :      // 上下キーのチェック
128 :      if( d & DS_KEY_UP ) {
129 :          m1_pwm = 50;
130 :      } else if( d & DS_KEY_DOWN ) {
131 :          m1_pwm = -50;
132 :      } else {
133 :          m1_pwm = 0;
134 :      }
135 :
136 :      // ジョイスティック  左の前後
137 :      m2_pwm = getDSJoyStickData(DS_LEFT_Y); // -100～100
138 :
139 :      // ジョイスティック  右の前後
140 :      m3_pwm = getDSJoyStickData(DS_RIGHT_X); // -100～100
141 :
142 :
143 :      /* 振動モータ小  制御 */
144 :      if( getDSKeyData() & DS_KEY_MARU ) {    // ○キー
145 :          ds_mini = 1;
146 :      } else {
147 :          ds_mini = 0;
148 :      }
149 :      /* 振動モータ大  制御 */
150 :      if( getDSKeyData() & DS_KEY_SHIKAKU ) { // □キー
151 :          ds_big = 255;
152 :      } else {
153 :          ds_big = 0;
154 :      }
155 :      setVaibrateData( ds_mini, ds_big );
156 :
157 :
158 :      }   // whileの終了
159 :  }
160 :
```

1 台目(親基板)は「**0**」、2 台目以降(子基板)は「**1**」を設定します。

コントローラのキー入力状態を液晶に表示します。液晶表示処理は、時間がかかるので 100ms に 1 回、表示します。
ロボット動作前は表示して、動作中は表示しないようにすると良いでしょう。

変数 d にコントローラのボタン情報を代入します。

「d & ボタン」で、ボタンが押されているかチェックします。
ボタンは、「scph_10010_r8c.h」の 4～19 行目に定義されていますので、コピペして使ってください。

モータの回転数(-100～100)を設定します。番号は 1～5 で、変数とモータの関係は下記の通りです。

m1_pwm … モータ 1(CN10)の PWM
m2_pwm … モータ 2(CN11)の PWM
m3_pwm … モータ 3(CN12)の PWM
m4_pwm … モータ 4(CN13)の PWM
m5_pwm … モータ 5(CN14)の PWM

最初の引数(今回は ds_mini)が、小モータの振動を制御します。0 で OFF、1 で ON です。
2 つめの引数(今回は ds_big)が、大モータの振動を制御します。0～255 を代入し 0～100%で振動します。

```
161 :  /************************************************************************/
162 :  /* R8C/38A スペシャルファンクションレジスタ(SFR)の初期化                */
163 :  /************************************************************************/
164 :  void init( void )
165 :  {
166 :      int     i;
167 :
168 :      /* クロックをXINクロック(20MHz)に変更 */
169 :      prc0  = 1;                          /* プロテクト解除               */
170 :      cm13  = 1;                          /* P4_6,P4_7をXIN-XOUT端子にする*/
171 :      cm05  = 0;                          /* XINクロック発振              */
172 :      for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約10ms) */
173 :      ocd2  = 0;                          /* システムクロックをXINにする  */
174 :      prc0  = 0;                          /* プロテクトON                 */
175 :
176 :      /* ポートの入出力設定 */
177 :
178 :      /*  PWM(予備)       左前M_PMW       右前M_PWM       ブザー
179 :          センサ左端      センサ左中      センサ右中      センサ右端  */
180 :      p0    = 0x00;
181 :      prc2 = 1;                           /* PD0のプロテクト解除          */
182 :      pd0  = 0xf0;
183 :
184 :      /*  センサ中心      ｽﾀｰﾄﾊﾞｰ         RxD0            TxD0
185 :          DIPSW3          DIPSW2          DIPSW1          DIPSW0          */
186 :      pur0 |= 0x04;                       /* P1_3～P1_0のプルアップON     */
187 :      p1  = 0x00;
188 :      pd1 = 0x10;
189 :
190 :      /*  右前M_方向      ステアM_方向    ステアM_PWM     右後M_PWM
191 :          右後M_方向      左後M_PWM       左後M_方向      左前M_方向      */
192 :      p2  = 0x00;
193 :      pd2 = 0xff;
194 :
195 :      /*  DS_CMD          none            DS_CLK          DS_DAT
196 :          DS_SEL          エンコーダB相   none            エンコーダA相   */
197 :      p3  = 0x00;
198 :      pd3 = 0x00;
199 :
200 :      /*  XOUT            XIN             ボード上のLED   none
201 :          none            VREF            none            none            */
202 :      p4  = 0x20;                         /* P4_5のLED:初期は点灯         */
203 :      pd4 = 0xb8;
204 :
205 :      /*  SW4             液晶E           液晶R/W         液晶RS
206 :          SW3or液晶D3     SW2or液晶D2     SW1or液晶D1     SW0or液晶D0     */
207 :      p5  = 0x00;
208 :      pd5 = 0x7f;
209 :
210 :      /*  none            none            none            none
211 :          none            none            none            none            */
212 :      p6  = 0x00;
213 :      pd6 = 0x00;
214 :
215 :      /*  CN6.2入力       CN6.3入力       CN6.4入力       CN6.5入力
216 :          none(ｱﾅﾛｸﾞ予備) 角度VR          センサ_左ｱﾅﾛｸﾞ  センサ_右ｱﾅﾛｸﾞ  */
217 :      p7  = 0x00;
218 :      pd7 = 0x00;
219 :
220 :      /*  DIPSWorLED      DIPSWorLED      DIPSWorLED      DIPSWorLED
221 :          DIPSWorLED      DIPSWorLED      DIPSWorLED      DIPSWorLED      */
222 :      pur2 |= 0x03;                       /* P8_7～P8_0のプルアップON     */
223 :      p8  = 0x00;
224 :      pd8 = 0x00;
225 :
226 :      /*  -               -               ﾌﾟｯｼｭｽｲｯﾁ       P8制御(LEDorSW)
227 :          右前M_Free      左前M_Free      右後M_Free      左後M_Free      */
228 :      p9  = 0x00;
229 :      pd9 = 0x1f;
230 :      pu23 = 1;                           /* P9_4,P9_5をプルアップする    */
231 :
232 :      /* A/Dコンバータの設定 */
233 :      admod   = 0x33;                     /* 繰り返し掃引モードに設定     */
234 :      adinsel = 0xb0;                     /* 入力端子P7の8端子を選択      */
235 :      adcon1  = 0x30;                     /* A/D動作可能                  */
236 :      asm(" nop ");                       /* φADの1サイクルウエイト入れる*/
237 :      adcon0  = 0x01;                     /* A/D変換スタート              */
238 :
239 :      /* タイマRBの設定 */
240 :      /* 割り込み周期 = 1 / 20[MHz]   * (TRBPRE+1) * (TRBPR+1)
241 :                      = 1 / (20*10^6) * 50         * 100
242 :                      = 0.00025[s] = 0.25[ms]
243 :      */
244 :      trbmr  = 0x00;                      /* 動作モード、分周比設定       */
245 :      trbpre =  50-1;                     /* プリスケーラレジスタ         */
246 :      trbpr  = 100-1;                     /* プライマリレジスタ           */
247 :      trbic  = 0x06;                      /* 割り込み優先レベル設定       */
248 :      trbcr  = 0x01;                      /* カウント開始                 */
249 :
```

タイマ RB を使って、0.25ms ごとに割り込みを発生させるようにします。
0.25ms ごとにコントローラ処理をします。

```
250 :       /* タイマRC PWMモード設定(左前モータ、右前モータ) */
251 :       trcpsr0 = 0x40;                     /* TRCIOA,B端子の設定           */
252 :       trcpsr1 = 0x33;                     /* TRCIOC,D端子の設定           */
253 :       trcmr   = 0x0f;                     /* PWMモード選択ビット設定      */
254 :       trccr1  = 0x8e;                     /* ｿｰｽｶｳﾝﾄ:f1,初期出力の設定    */
255 :       trccr2  = 0x00;                     /* 出力レベルの設定             */
256 :       trcgra  = TRC_MOTOR_CYCLE - 1;      /* 周期設定                     */
257 :       trcgrb  = trcgrb_buff = trcgra;     /* P0_5端子のON幅(左前モータ)   */
258 :       trcgrc  = trcgra;                   /* P0_7端子のON幅(予備)         */
259 :       trcgrd  = trcgrd_buff = trcgra;     /* P0_6端子のON幅(右前モータ)   */
260 :       trcic   = 0x07;                     /* 割り込み優先レベル設定       */
261 :       trcier  = 0x01;                     /* IMIAを許可                   */
262 :       trcoer  = 0x01;                     /* 出力端子の選択               */
263 :       trcmr  |= 0x80;                     /* TRCカウント開始              */
264 :
265 :       /* タイマRD リセット同期PWMモード設定(左後ﾓｰﾀ、右後ﾓｰﾀ、ｻｰﾎﾞﾓｰﾀ) */
266 :       trdpsr0 = 0x08;                     /* TRDIOB0,C0,D0端子設定        */
267 :       trdpsr1 = 0x05;                     /* TRDIOA1,B1,C1,D1端子設定     */
268 :       trdmr   = 0xf0;                     /* バッファレジスタ設定         */
269 :       trdfcr  = 0x01;                     /* リセット同期PWMモードに設定  */
270 :       trdcr0  = 0x20;                     /* ソースカウントの選択:f1      */
271 :       trdgra0 = trdgrc0 = TRD_MOTOR_CYCLE - 1;    /* 周期設定             */
272 :       trdgrb0 = trdgrd0 = 0;              /* P2_2端子のON幅(左後モータ)   */
273 :       trdgra1 = trdgrc1 = 0;              /* P2_4端子のON幅(右後モータ)   */
274 :       trdgrb1 = trdgrd1 = 0;              /* P2_5端子のON幅(サーボモータ) */
275 :       trdoer1 = 0xcd;                     /* 出力端子の選択               */
276 :       trdstr  = 0x0d;                     /* TRD0カウント開始             */
277 :
278 :   #if 1
279 :       // ifを1にすると、この中を実行
280 :       /* タイマRG タイマモード(両エッジでカウント)の設定 */
281 :       timsr = 0x40;                       /* TRGCLKA端子 P3_0に割り当てる */
282 :       trgcr = 0x15;                       /* TRGCLKA端子の両エッジでカウント*/
283 :       trgmr = 0x80;                       /* TRGのカウント開始            */
284 :   #else
285 :       // ifを0にすると、この中を実行
286 :       /* タイマRG(位相計数モード【A相,B相で正転,逆転検出】 の設定 */
287 :       timsr = 0xc0;                       /* TRGCLKA,TRGCLKB端子割り当て  */
288 :       trgcntc = 0xff;                     /* 位相計数ﾓｰﾄﾞのｶｳﾝﾄ方法指定   */
289 :       trgmr = 0x82;                       /* TRGのカウント開始            */
290 :   #endif
291 :   }
292 :
293 :   /************************************************************************/
294 :   /* タイマRB 0.25msごとの割り込み処理                                    */
295 :   /************************************************************************/
296 :   #pragma interrupt /B intTRB(vect=24)
297 :   void intTRB( void )
298 :   {
299 :       static int trb_cnt = 0;             /* 1msカウント用                */
300 :       static int iTimer10 = 0;            /* 10msカウント用               */
301 :       unsigned int i;
302 :
303 :       trb_cnt++;
304 :
305 :       asm(" fset I ");                    /* タイマRB以上の割り込み許可   */
306 :
307 :       /* コントローラ処理 */
308 :       dsProcess();
309 :
310 :       if( trb_cnt >= 4 ) {
311 :           /* 1msごとに実行 */
312 :           trb_cnt = 0;
313 :
314 :           cnt1++;
315 :           cnt_lcd++;
316 :
317 :           /* モータ制御 */
318 :           motor2_r( m1_pwm, m2_pwm );
319 :           servoPwmOut( m3_pwm );
320 :           motor2_f( m4_pwm, m5_pwm );
321 :
322 :           /* 液晶表示処理用関数(1msごとに実行) */
323 :           lcdShowProcess();
324 :
325 :           /* ブザー処理 */
326 :           beepProcessS();
327 :
328 :           /* 10回中1回実行する処理 */
329 :           iTimer10++;
330 :           switch( iTimer10 ) {
331 :           case 1:
332 :               /* タイマRGのエンコーダ制御 */
333 :               i = trg;
334 :               iEncoder       = i - uEncoderBuff;
335 :               lEncoderTotal += iEncoder;
336 :               uEncoderBuff = i;
337 :               break;
338 :
```

1 にすると、ロータリエンコーダのパルスを 1 相分、マイコンに入力します。接続は、P3_0 端子になります(モータドライブ基板 TypeS Ver.4 の CN9 の 2 ピン)。
0 にすると、ロータリエンコーダのパルスを 2 相分、マイコンに入力します。接続は、P3_0 端子と P3_2 端子になります。P3_2 端子はモータドライブ基板 TypeS Ver.4 のコネクタからは出力されていないので、マイコンボードの CN5 の 7 ピンと直接接続してください。

コントローラと 0.25ms ごとに通信をして、ボタン情報などを入力します。

0.25ms ごとにプログラムを実行すると、マイコンの負荷が多くなりすぎてプログラムの実行が追いつかなくなることがあるので、コントローラ処理以外は、1ms ごとに実行します。

5 個のモータを PWM をセットするのは割り込み内で行っています。

```
339 :          case 2:
340 :              /* スイッチ読み込み準備 */
341 :              p9_4 = 0;                       /* LED出力OFF                   */
342 :              pd8  = 0x00;
343 :              break;
344 :
345 :          case 3:
346 :              /* スイッチ読み込み、LED出力 */
347 :              types_dipsw = ~p8;          /* ﾄﾞﾗｲﾌﾞ基板TypeS Ver.4のSW読み込み*/
348 :              p8  = types_led;            /* ﾄﾞﾗｲﾌﾞ基板TypeS Ver.4のLEDへ出力 */
349 :              pd8 = 0xff;
350 :              p9_4 = 1;                       /* LED出力ON                    */
351 :              break;
352 :
353 :          case 4:
354 :              break;
355 :
356 :          case 5:
357 :              break;
358 :
359 :          case 6:
360 :              break;
361 :
362 :          case 7:
363 :              break;
364 :
365 :          case 8:
366 :              break;
367 :
368 :          case 9:
369 :              break;
370 :
371 :          case 10:
372 :              /* iTimer10変数の処理 */
373 :              iTimer10 = 0;
374 :              break;
375 :          }
376 :      }
377 :  }
378 :
379 :  /************************************************************************/
380 :  /* タイマRC 割り込み処理                                                */
381 :  /************************************************************************/
382 :  #pragma interrupt intTRC(vect=7)
383 :  void intTRC( void )
384 :  {
385 :      trcsr &= 0xfe;
386 :
387 :      /* タイマRC  デューティ比の設定 */
388 :      trcgrb = trcgrb_buff;
389 :      trcgrd = trcgrd_buff;
390 :  }
391 :
392 :  /************************************************************************/
393 :  /* マイコンボード上のディップスイッチ値読み込み                         */
394 :  /* 引数   なし                                                          */
395 :  /* 戻り値 スイッチ値 0～15                                              */
396 :  /************************************************************************/
397 :  unsigned char dipsw_get( void )
398 :  {
399 :      unsigned char sw;
400 :
401 :      sw = p1 & 0x0f;                     /* P1_3～P1_0読み込み           */
402 :
403 :      return sw;
404 :  }
405 :
406 :  /************************************************************************/
407 :  /* モータドライブ基板TypeS Ver.4上のディップスイッチ値読み込み          */
408 :  /* 引数   なし                                                          */
409 :  /* 戻り値 スイッチ値 0～255                                             */
410 :  /************************************************************************/
411 :  unsigned char dipsw_get2( void )
412 :  {
413 :      /* 実際の入力はタイマRB割り込み処理で実施 */
414 :      return types_dipsw;
415 :  }
416 :
417 :  /************************************************************************/
418 :  /* モータドライブ基板TypeS Ver.4上のプッシュスイッチ値読み込み          */
419 :  /* 引数   なし                                                          */
420 :  /* 戻り値 スイッチ値 0:OFF 1:ON                                         */
421 :  /************************************************************************/
422 :  unsigned char pushsw_get( void )
423 :  {
424 :      unsigned char sw;
425 :
426 :      sw = ~p9_5 & 0x01;
427 :
428 :      return sw;
429 :  }
```

```
430 :
431 :  /************************************************************************/
432 :  /* モータドライブ基板TypeS Ver.4のCN6の状態読み込み                      */
433 :  /* 引数   なし                                                          */
434 :  /* 戻り値 0～15                                                         */
435 :  /************************************************************************/
436 :  unsigned char cn6_get( void )
437 :  {
438 :      unsigned char data;
439 :
440 :      data = p7 >> 4;
441 :
442 :      return data;
443 :  }
444 :
445 :  /************************************************************************/
446 :  /* モータドライブ基板TypeS Ver.4のLED制御                               */
447 :  /* 引数   8個のLED制御 0:OFF 1:ON                                       */
448 :  /* 戻り値 なし                                                          */
449 :  /************************************************************************/
450 :  void led_out( unsigned char led )
451 :  {
452 :      /* 実際の出力はタイマRB割り込み処理で実施 */
453 :      types_led = led;
454 :  }
455 :
456 :  /************************************************************************/
457 :  /* 後輪の速度制御2 ディップスイッチには関係しないmotor関数              */
458 :  /* 引数   左モータ:-100～100 , 右モータ:-100～100                       */
459 :  /*        0で停止、100で正転100%、-100で逆転100%                        */
460 :  /* 戻り値 なし                                                          */
461 :  /************************************************************************/
462 :  void motor2_r( int accele_l, int accele_r )
463 :  {
464 :      /* 左後モータ */
465 :      if( accele_l >= 0 ) {
466 :          p2_1 = 0;
467 :          trdgrd0 = (long)( TRD_MOTOR_CYCLE - 2 ) * accele_l / 100;
468 :      } else {
469 :          p2_1 = 1;
470 :          trdgrd0 = (long)( TRD_MOTOR_CYCLE - 2 ) * ( -accele_l ) / 100;
471 :      }
472 :
473 :      /* 右後モータ */
474 :      if( accele_r >= 0 ) {
475 :          p2_3 = 0;
476 :          trdgrc1 = (long)( TRD_MOTOR_CYCLE - 2 ) * accele_r / 100;
477 :      } else {
478 :          p2_3 = 1;
479 :          trdgrc1 = (long)( TRD_MOTOR_CYCLE - 2 ) * ( -accele_r ) / 100;
480 :      }
481 :  }
482 :
483 :  /************************************************************************/
484 :  /* 前輪の速度制御2 ディップスイッチには関係しないmotor関数              */
485 :  /* 引数   左モータ:-100～100 , 右モータ:-100～100                       */
486 :  /*        0で停止、100で正転100%、-100で逆転100%                        */
487 :  /* 戻り値 なし                                                          */
488 :  /************************************************************************/
489 :  void motor2_f( int accele_l, int accele_r )
490 :  {
491 :      /* 左前モータ */
492 :      if( accele_l >= 0 ) {
493 :          p2_0 = 0;
494 :      } else {
495 :          p2_0 = 1;
496 :          accele_l = -accele_l;
497 :      }
498 :      if( accele_l <= 5 ) {
499 :          trcgrb = trcgrb_buff = trcgra;
500 :      } else {
501 :          trcgrb_buff = (unsigned long)(TRC_MOTOR_CYCLE-2) * accele_l / 100;
502 :      }
503 :
504 :      /* 右前モータ */
505 :      if( accele_r >= 0 ) {
506 :          p2_7 = 0;
507 :      } else {
508 :          p2_7 = 1;
509 :          accele_r = -accele_r;
510 :      }
511 :      if( accele_r <= 5 ) {
512 :          trcgrd = trcgrd_buff = trcgra;
513 :      } else {
514 :          trcgrd_buff = (unsigned long)(TRC_MOTOR_CYCLE-2) * accele_r / 100;
515 :      }
516 :  }
517 :
```

```
518 :  /************************************************************************/
519 :  /* 後モータ停止動作 (ブレーキ、フリー)                                  */
520 :  /* 引数   左モータ:FREE or BRAKE , 右モータ:FREE or BRAKE               */
521 :  /* 戻り値 なし                                                          */
522 :  /************************************************************************/
523 :  void motor_mode_r( int mode_l, int mode_r )
524 :  {
525 :      if( mode_l ) {
526 :          p9_0 = 1;
527 :      } else {
528 :          p9_0 = 0;
529 :      }
530 :      if( mode_r ) {
531 :          p9_1 = 1;
532 :      } else {
533 :          p9_1 = 0;
534 :      }
535 :  }
536 :
537 :  /************************************************************************/
538 :  /* 前モータ停止動作 (ブレーキ、フリー)                                  */
539 :  /* 引数   左モータ:FREE or BRAKE , 右モータ:FREE or BRAKE               */
540 :  /* 戻り値 なし                                                          */
541 :  /************************************************************************/
542 :  void motor_mode_f( int mode_l, int mode_r )
543 :  {
544 :      if( mode_l ) {
545 :          p9_2 = 1;
546 :      } else {
547 :          p9_2 = 0;
548 :      }
549 :      if( mode_r ) {
550 :          p9_3 = 1;
551 :      } else {
552 :          p9_3 = 0;
553 :      }
554 :  }
555 :
556 :  /************************************************************************/
557 :  /* サーボモータ制御                                                     */
558 :  /* 引数   サーボモータPWM:-100～100                                     */
559 :  /*        0で停止、100で正転100%、-100で逆転100%                        */
560 :  /* 戻り値 なし                                                          */
561 :  /************************************************************************/
562 :  void servoPwmOut( int pwm )
563 :  {
564 :      if( pwm >= 0 ) {
565 :          p2_6 = 0;
566 :          trdgrd1 = (long)( TRD_MOTOR_CYCLE - 2 ) * pwm / 100;
567 :      } else {
568 :          p2_6 = 1;
569 :          trdgrd1 = (long)( TRD_MOTOR_CYCLE- 2 ) * ( -pwm ) / 100;
570 :      }
571 :  }
572 :
573 :  /************************************************************************/
574 :  /* end of file                                                          */
575 :  /************************************************************************/
```

4.3 R8C/38A マイコンで使用する内蔵周辺機能

機能	詳細
SSU	コントローラと信号のやり取りを行います。 2 台目以降の場合は、1 台目からの信号を受信します。
タイマ RB	0.25[ms]ごとの割り込み用として使用しています。
タイマ RC	モータドライブ基板 TypeS Ver.4 の左前モータ(CN13)、右前モータ(CN14)のモータを制御しています。
タイマ RD	リセット同期 PWM モードとして使用して、左後モータ(CN10)、右後モータ(CN11)、サーボモータ(CN12)のモータを制御します。
タイマ RG	P3_0 端子からのパルスカウントとして使用します。立ち上がり、立ち下がりの両方でカウントします。ロボットに使うと、移動速度や、移動距離が分かりますが、前進しているか後進しているかは分かりません。 位相計数モードというモードに設定し、ロータリエンコーダのＡ相、Ｂ相の信号を P3_0 端子、P3_2 端子に入力すると、正転（前進）、逆転（後進）が分かります。

5. 大電流モータを接続する

5.1 FET の選定

モータドライブ基板 TypeS Ver.4 は、マイコンカーラリー承認の指定モータは駆動できますが、それ以上電流の流れるモータは FET の容量が足りず FET が焼けてしまいます。
キットに入っている FET と、市販されていて交換できそうな FET を下表に示します。

	P チャンネル(2SJ タイプ)	N チャネル(2SK タイプ)
キットの FET	2SJ530(L) ドレイン-ソース電圧:60V ドレイン電流:15A	2SK2869(L) ドレイン-ソース電圧:60V ドレイン電流:20A
秋月電子通商	2SJ471 ドレイン-ソース電圧:30V ドレイン電流:30A http://akizukidenshi.com/catalog/g/gI-00034/	H7N0308CF ドレイン-ソース電圧:30V ドレイン電流:60A http://akizukidenshi.com/catalog/g/gI-01110/

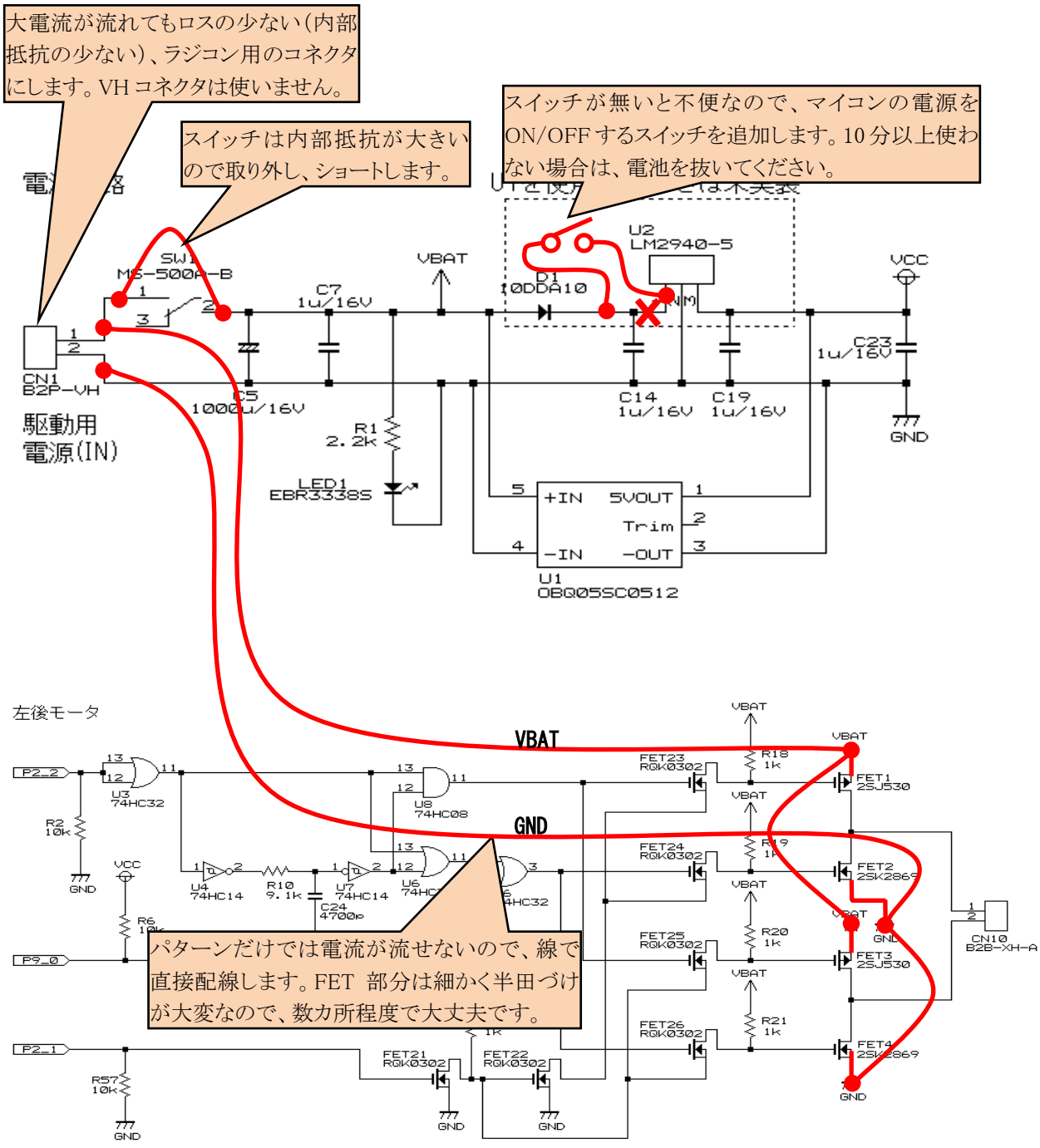
選定方法は、ピンの順番が「ゲート(G)、ドレイン(D)、ソース(S)」で、ピン間が 2.54mm であればほとんどの FET は使用できます。

5.2 取り付け方法

モータドライブ基板 TypeS Ver.4 は、FET と FET の実装間隔が狭く TO-220 パッケージの FET を取り付けることができません。そのため、苦肉の策ですが FET を 2SJ タイプは部品面から取り付け、2SK タイプは半田面から取り付けると基板に実装することができます。もちろん、半田面に FET を取り付けることになりますので、ロボットに固定するときは、空きスペースを確保してください。

5.3 その他の改造ポイント

モータドライブ基板 TypeS Ver.4 の FET 交換以外の改造ポイントを、下記に示します。



6. 参考文献

・ルネサス エレクトロニクス(株)
R8C/38C グループ　ユーザーズマニュアル　ハードウェア編 Rev.1.10
・ルネサス エレクトロニクス(株)
M16C シリーズ,R8C ファミリ用 C/C++コンパイラパッケージ V.6.00
C/C++コンパイラユーザーズマニュアル Rev.1.00
・ルネサス エレクトロニクス(株)
High-performance Embedded Workshop V.4.09　ユーザーズマニュアル Rev.1.00
・ルネサス半導体トレーニングセンター　C言語入門コーステキスト　第 1 版
・電波新聞社　マイコン入門講座　大須賀威彦著　第 1 版
・ソフトバンク(株)　新C言語入門シニア編　林晴比古著　初版
・共立出版(株)　プログラマのための ANSI C 全書　L.Ammeraal 著
吉田敬一・竹内淑子・吉田恵美子訳　初版

・プレイステーション・PAD／メモリ・インターフェースの解析　Nifty－ID:HFB03536　藤田
http://kaele.com/~kashima/games/ps_jpn.txt
・デュアルショック 2(SCPH-10010)の新機能使用方法
http://applause.elfmimi.jp/dualshock/millar/NT/dualshock_2.txt

マイコンカーラリー、販売部品についての詳しい情報は、マイコンカーラリー販売サイトをご覧ください。

https://www2.himdx.net/mcr/

R8C マイコンについての詳しい情報は、ルネサス エレクトロニクス(株)のホームページをご覧ください。

http://japan.renesas.com/

の製品情報にある「マイコン」→「R8C」でご覧頂けます