マイコンカーキットVer.5
動作確認マニュアル
R8C/38A版

本マニュアルは、
・モータドライブ基板 Ver.4
・センサ基板 Ver.4.1
に対応したマイコンカーキット Ver.5 の動作確認手順を説明しています。

第 1.01 版
2012.03.21
ジャパンマイコンカーラリー実行委員会

注 意 事 項 (rev.3.0J)

著作権

・本マニュアルに関する著作権はジャパンマイコンカーラリー実行委員会に帰属します。
・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、文書によるジャパンマイコンカーラリー実行委員会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、ジャパンマイコンカーラリー実行委員会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、ジャパンマイコンカーラリー実行委員会は、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。

連絡先

(株)ルネサスソリューションズ　ルネサスマイコンカーラリー事務局
〒162-0824　東京都新宿区揚場町 2-1　軽子坂MNビル
TEL　(03)-3266-8510
E-mail:official@mcr.gr.jp

目　次

1. 概要

本マニュアルは、下記マニュアルで製作、組み立てたマイコンカーの動作確認方法を説明するマニュアルです。

●モータドライブ基板 Ver.4 製作マニュアル
●センサ基板 Ver.4.1 製作マニュアル
●マイコンカーキット Ver.5 本体組み立て製作マニュアル

動作確認をするためには、下記の手順で行います。

1. パソコンにルネサス統合開発環境をインストールします
パソコンにルネサス統合開発環境(M16C、R8C 版)をインストールします。詳しくは、「ルネサス統合開発環境操作マニュアル(R8C/38A 版)」を参照して、インストールしてください。**なお、ルネサス統合開発環境(H8 版)では動作しません。**

2. サンプルプログラム(動作確認プログラム)をインストールします
パソコンに R8C/38A 用のサンプルプログラム(workspace_r8c38a_100.exe、100 の部分はバージョンにより異なります)をインストールします。詳しくは、「ルネサス統合開発環境操作マニュアル(R8C/38A 版)」を参照して、インストールしてください。

3. 動作確認プログラムを、マイコンカーの RY_R8C38 ボードに書き込みます
これから行います。

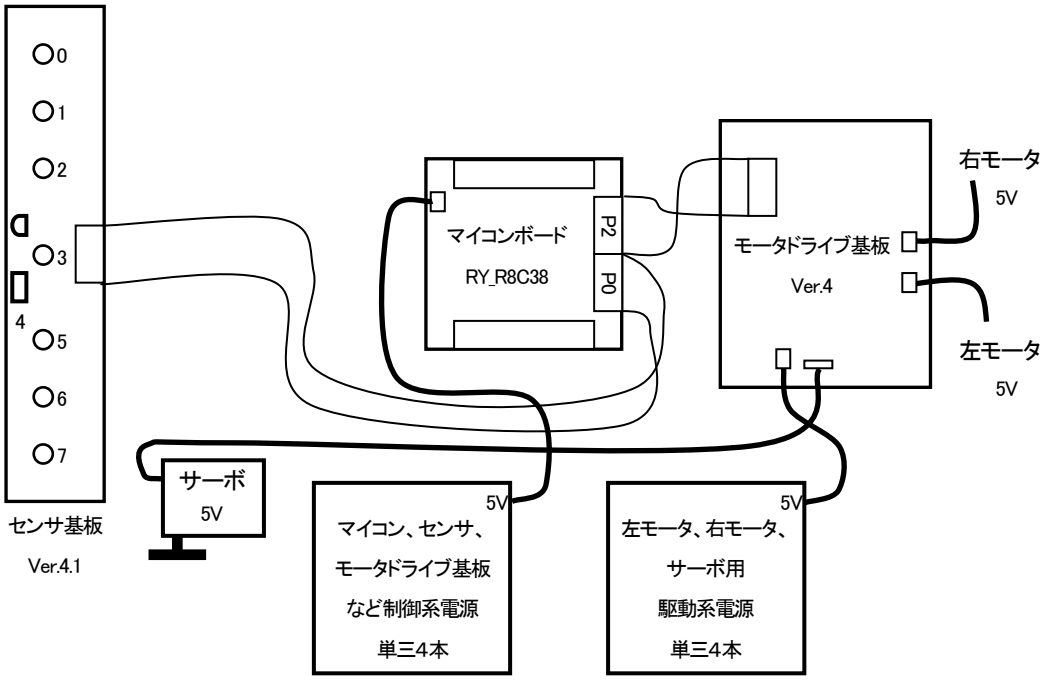
4. 本マニュアルに従って動作確認します
これから行います。

5. サーボセンタと最大切れ角を調整します
マイコンカーキット Ver.5 kit07_38a プログラム解説マニュアル(R8C/38A 版)の「7. サーボセンタと最大切れ角の調整」を参照して、調整してください。

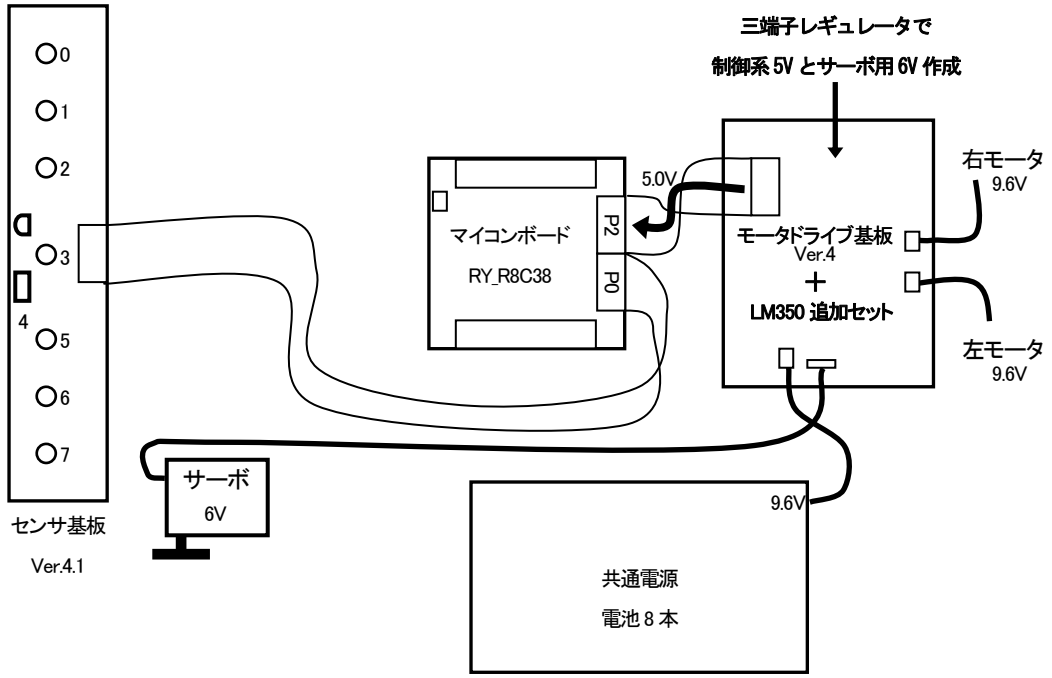
6. 走行プログラムを書き込みます
マイコンカーキット Ver.5 kit07_38a プログラム解説マニュアル(R8C/38A 版)を参照しながら、プロジェクト「kit07_38a」の kit07_38a.mot ファイルを RY_R8C38 ボードに書き込み、コースを走らせてみてください。

2. マイコンカーキットの構成

本マニュアルでは、マイコンカーキット Ver.5 の構成のマイコンカーの動作確認を行うことができます。構成を、下記に示します。

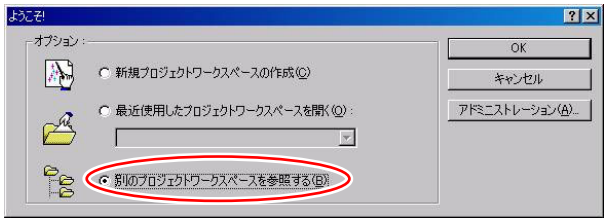
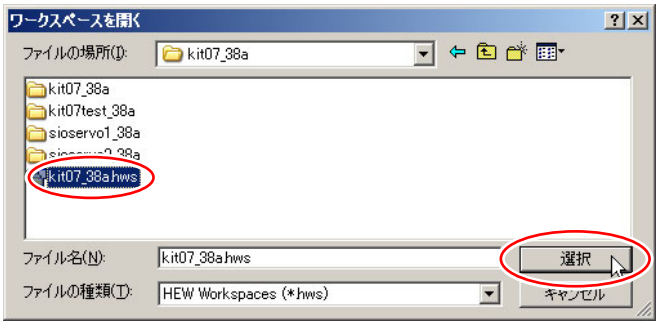
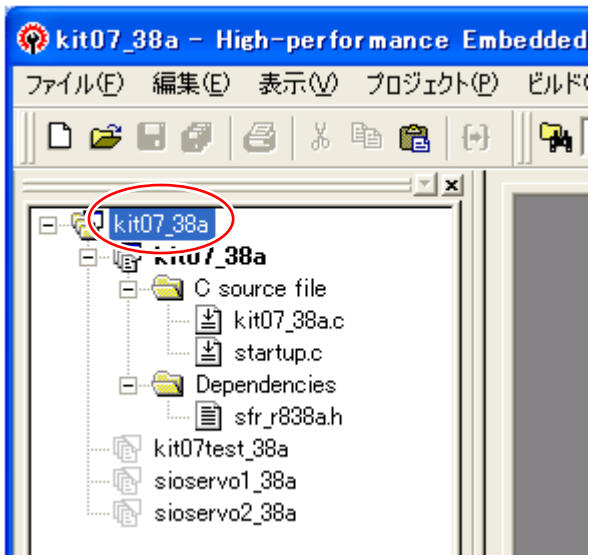


モータドライブ基板に「LM350 追加セット」を追加した構成のマイコンカーも、動作確認することができます。追加したときの構成を、下記に示します。

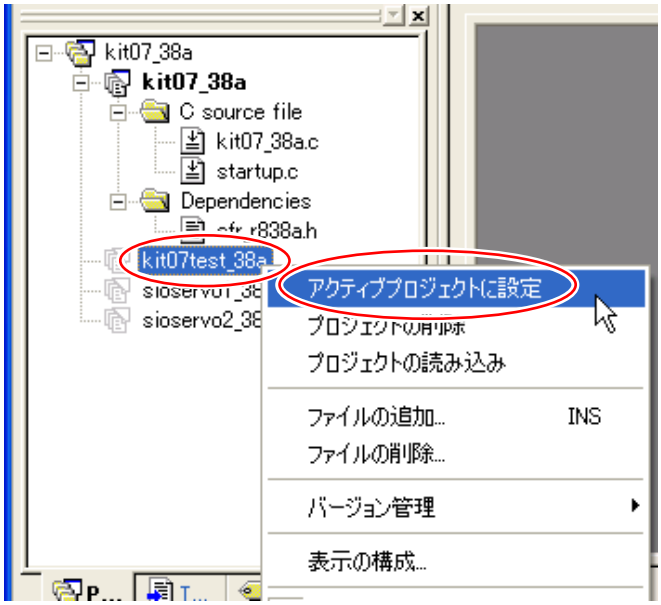
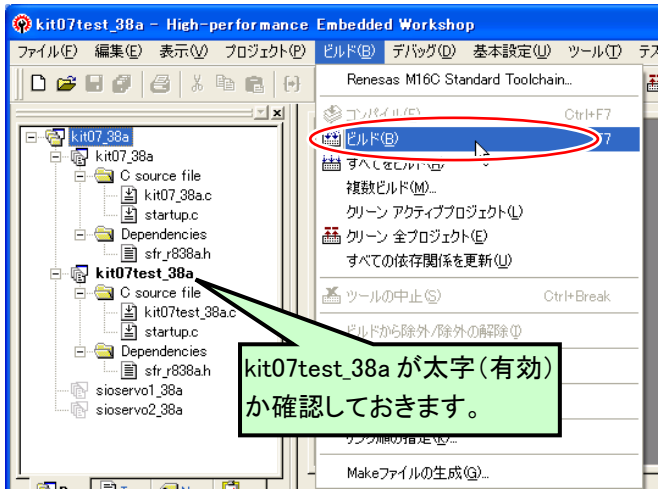
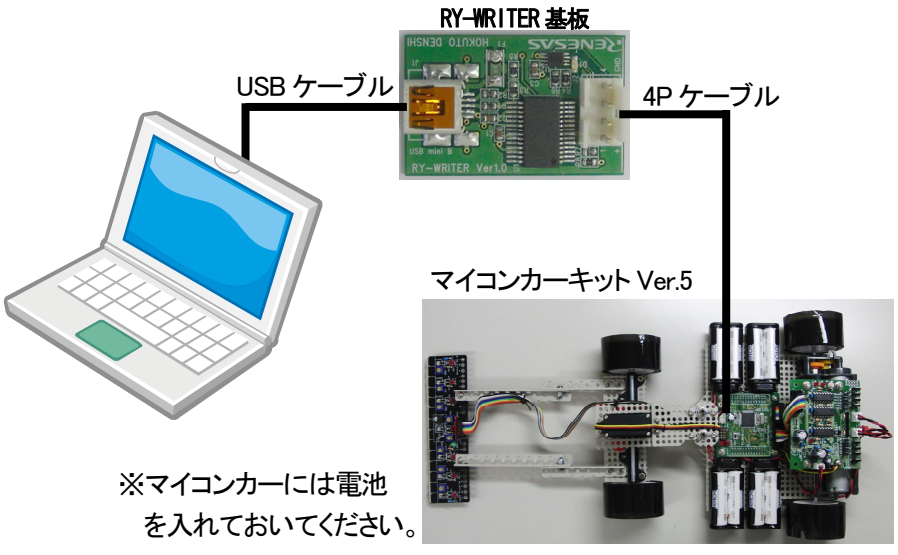


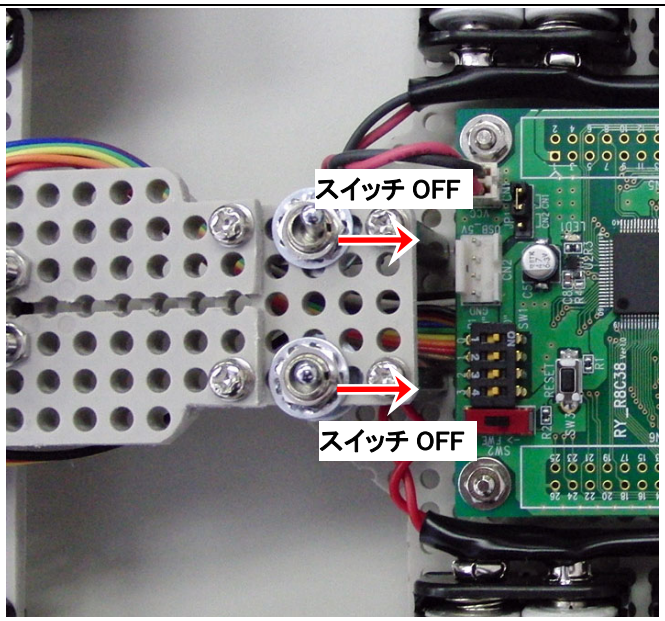
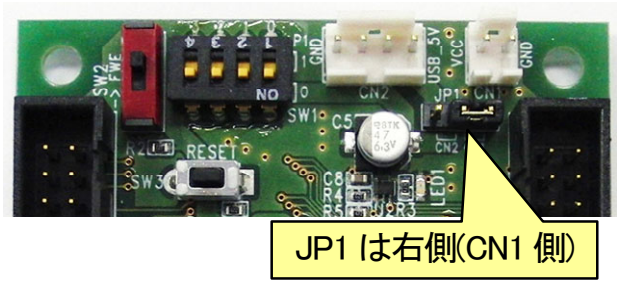
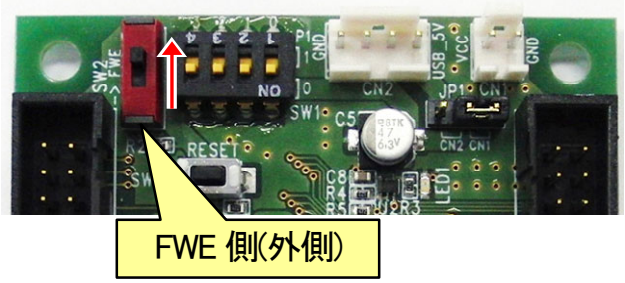
3. 動作確認用プログラムの書き込み

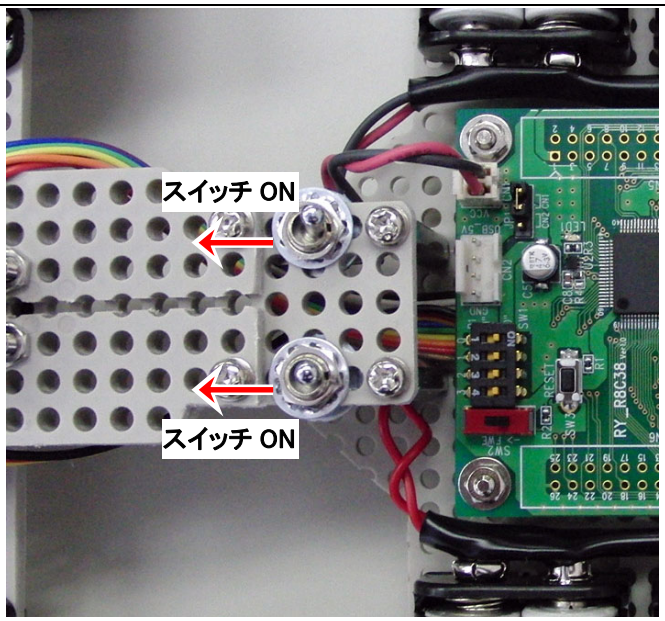
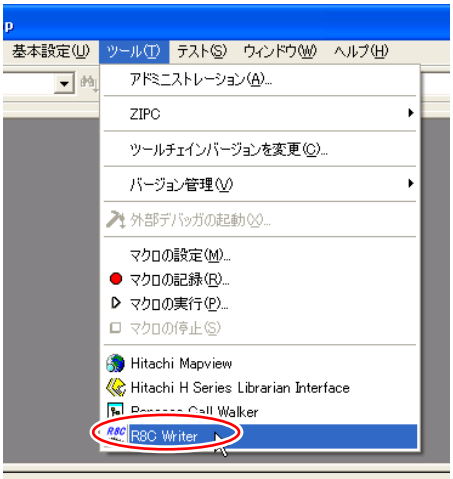
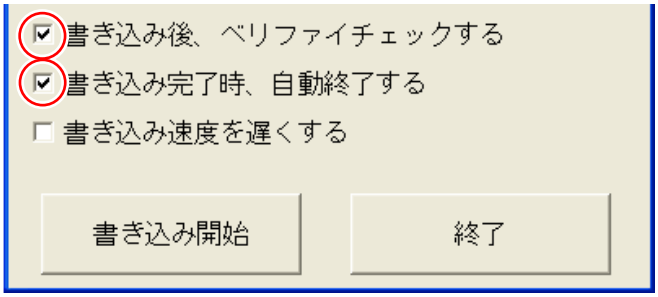
3.1 ワーススペース「kit07_38a」を開く

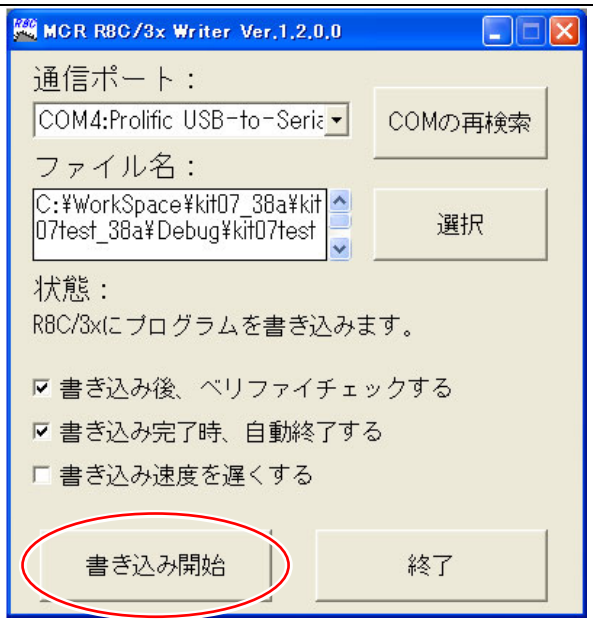
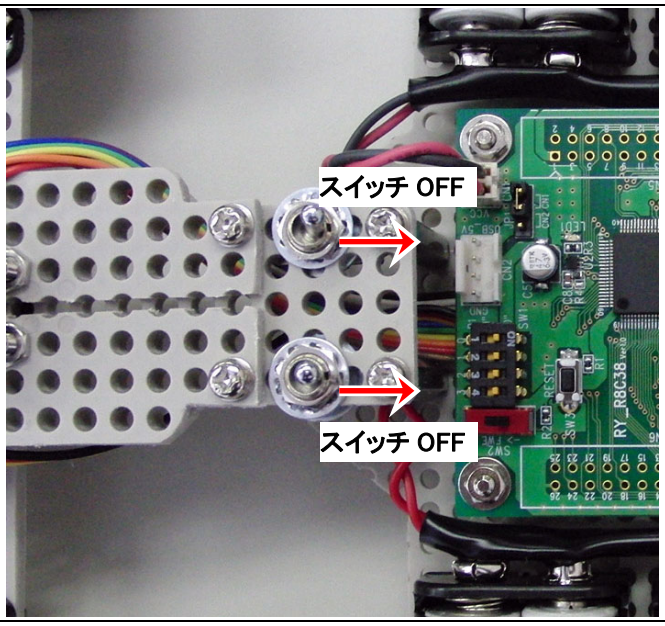
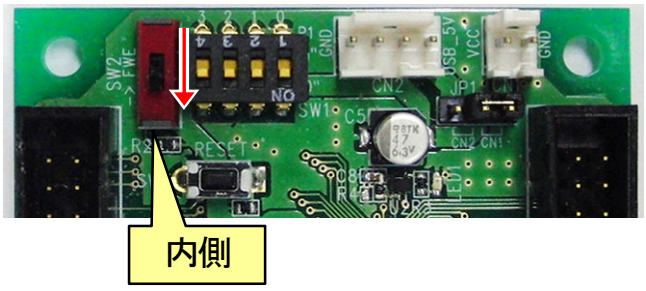
1		ルネサス統合開発環境を実行します。
2		「別のプロジェクトワークスペースを参照する」を選択します。
3		Cドライブ → Workspace → kit07_38a の「kit07_38a.hws」を選択します。
4		ワークスペース「kit07_38a」が開きます。このワークスペースには、4 つのプロジェクトがあります。 ●kit07_38a マイコンカー走行プログラムです。 **●kit07test_38a** **今回使うプロジェクトで、モータドライブ基板、センサ基板が正しく動作するか確認します。** ●sioservo1_38a サーボのセンタを調整するプログラムです。 ●sioservo2_38a サーボの最大切れ角を見つけるためのプログラムです。

3.2 動作確認用プログラムの書き込み

1		「kit07test_38a」をアクティブプロジェクトに設定します。 「kit07test_38a」上で右クリックします。 「アクティブプロジェクトに設定」を選択します。 「kit07test_38a」が太字になります。
2		「ビルド→ビルド」で MOT ファイルを作ります。
3		パソコンとマイコンカー(RY_R8C38 ボード)の間に USB 信号を TTL レベル信号に変換する基板を入れます。 左図は、RY-WRITER 基板をパソコンとマイコンカーキット Ver.5 の間に入れた場合の結線です。詳しくは、「マイコン実習マニュアル(R8C/38A 版)」を参照してください。

4		マイコンカーの電源スイッチは、2 つとも切っておきます。
5		RY_R8C38 ボードの JP1 は、写真のように右側をショート(CN1 側)してください。 ※CN1 から電源供給する設定です。CN2 にすると、パソコンの USB 電源から電源供給する設定になります。パソコンの USB 電源は、電流を取りすぎると壊れるおそれがあるので使わないでください。
6		SW2 を FWE 側(外側)にします。 SW2 を外側の状態で RY_R8C38 ボードの電源を入れると、マイコンはプログラム書き込みモードになります。 **※SW2 は、必ず RY_R8C38 ボードの電源が OFF の状態で操作してください。**

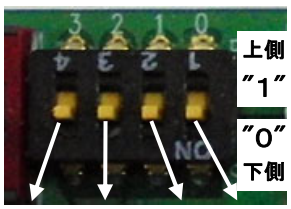
7		マイコンカーの電源スイッチを、2 つとも ON にします。
8		「ツール→R8C Writer」で R8C Writer を立ち上げます。 ※「R8C Writer」が無い場合は、「ルネサス統合開発環境操作マニュアル(R8C/38A 版」を参照して、登録してください。
9		R8C Writer のチェックボックスの設定を確認します。 2 カ所とも、チェックはON(☑)にしておきます。

10		「kit07test_38a.mot」を書き込みます。 書き込みが正常に終わったら、R8C Writer は自動で終了します。もしエラー画面が出てきた場合は、ケーブルの接続などを確認して再度書き込んでください。 ※Tera Term が立ち上がっていると、通信ポートを使うので書き込みができません。Tera Term を終了させて、再度書き込んでください。
11		マイコンカーの電源スイッチは、2 つとも切っておきます。
12		SW2 を内側にします。 SW2 を内側の状態で RY_R8C38 ボードの電源を入れると、先ほど書き込んだプログラムを実行します。 **※SW2 は、必ず RY_R8C38 ボードの電源が OFF の状態で操作してください。**

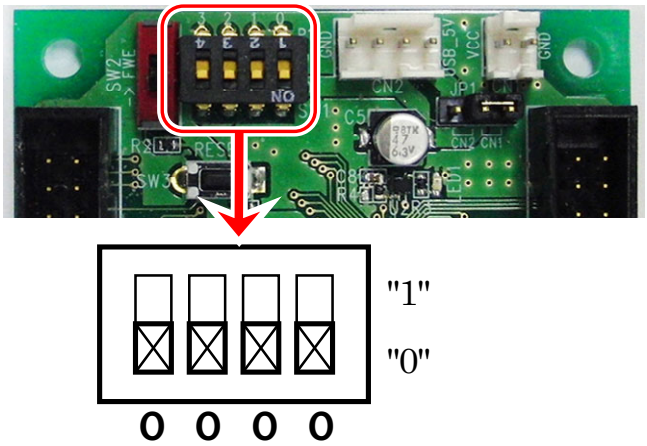
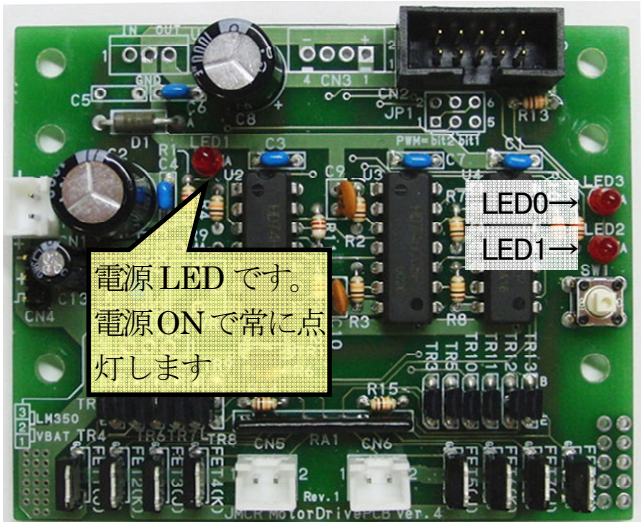
4. 動作確認

4.1 動作確認一覧

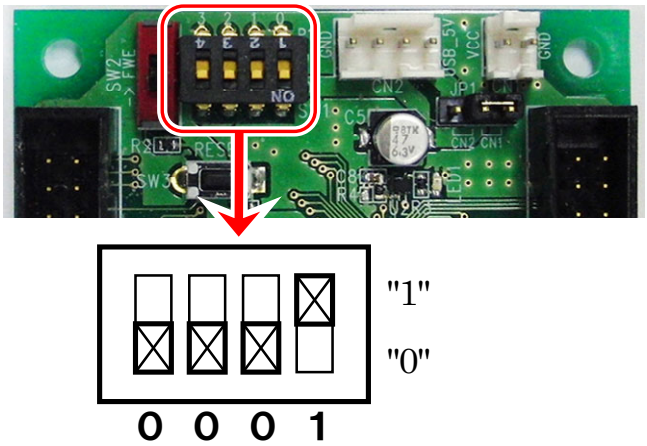
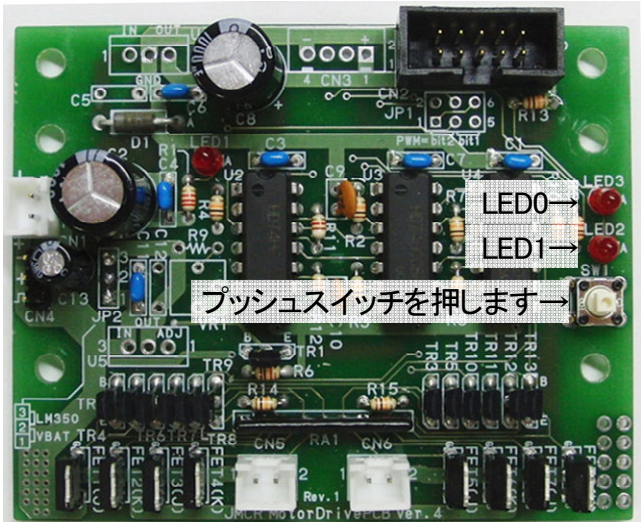
RY_R8C38 ボードのディップスイッチの状態を変更することにより、マイコンカーのどの部分を動作確認するか選択し、動作確認を行います。

				内容
P1_3	P1_2	P1_1	P1_0	
0	0	0	0	LED の動作確認をします。LED が 0.5 秒間隔で交互に点灯します。
0	0	0	1	プッシュスイッチの動作確認をします。 スイッチ OFF で LED0 が点灯、スイッチ ON で LED1 が点灯します。
0	0	1	0	サーボの動作確認をします。 サーボが、「0° →右 30° →左 30° の繰り返し」の動作をします。
0	0	1	1	何もしません。
0	1	0	0	右モータの動作確認をします。 「正転→ブレーキ」動作を繰り返します。
0	1	0	1	右モータの動作確認をします。 「逆転→ブレーキ」動作を繰り返します。
0	1	1	0	左モータの動作確認をします。 「正転→ブレーキ」動作を繰り返します。
0	1	1	1	左モータの動作確認をします。 「逆転→ブレーキ」動作を繰り返します。
1	0	0	0	センサ基板の bit1、bit0 の動作確認をします。 センサ bit1、bit0 の状態を LED1、LED0 に出力します。
1	0	0	1	センサ基板の bit3、bit2 の動作確認をします。 センサ bit3、bit2 の状態を LED1、LED0 に出力します。
1	0	1	0	センサ基板の bit5、bit4 の動作確認をします。 センサ bit5、bit4 の状態を LED1、LED0 に出力します。
1	0	1	1	センサ基板の bit7、bit6 の動作確認をします。 センサ bit7、bit6 の状態を LED1、LED0 に出力します。
1	1	0	0	まっすぐに走るか確認します。 PWM50%で前進、2 秒後に停止します。
1	1	0	1	まっすぐに走るか確認します。 PWM50%で前進、5 秒後に停止します。
1	1	1	0	まっすぐに走るか確認します。 PWM100%で前進、2 秒後に停止します。
1	1	1	1	まっすぐに走るか確認します。 PWM100%で前進、5 秒後に停止します。

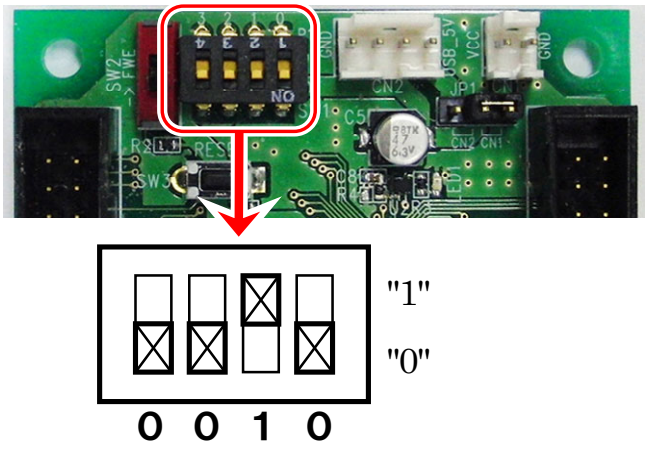
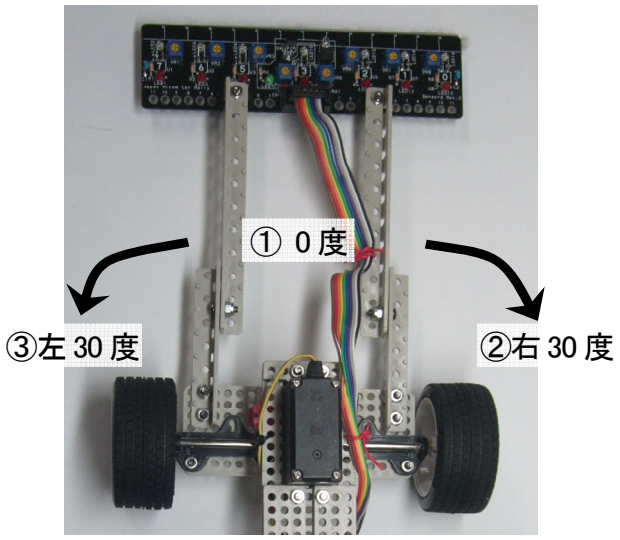
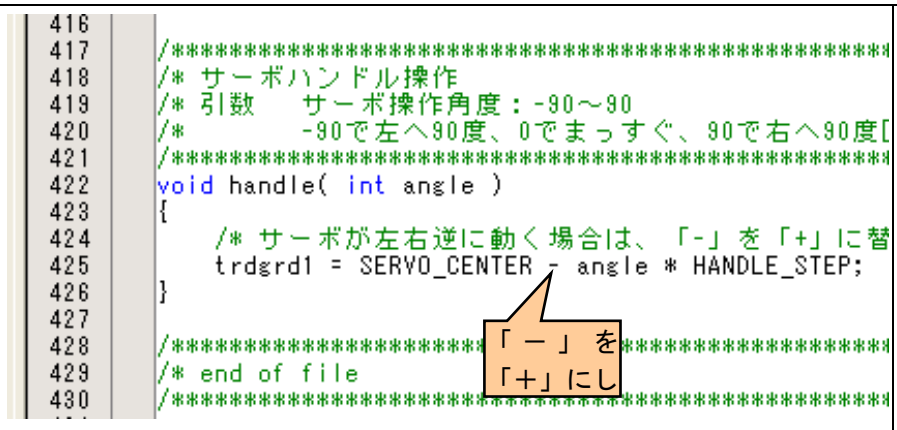
4.2 LEDの動作確認

1		モータドライブ基板の LED が点灯・消灯するか確認します。 ディップスイッチを"0000"の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。
2		モータドライブ基板の LED0 と LED1 が 0.5 秒毎に交互に点灯します。 動作確認が終わったら電源スイッチ 2 つを OFF にします。 LED0、または LED1 が点灯しない場合は、RY_R8C38 ボードとモータドライブ基板を接続するフラットケーブルの不良、LED の半田付け不良、半田ブリッジ(ショート)、LED の向きが逆など考えられます。目視チェック、テスタなどで原因を突き止めてください。

4.3 プッシュスイッチの動作確認

1		モータドライブ基板のプッシュスイッチが反応するか確認します。 ディップスイッチを"0001"の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。
2		モータドライブ基板上のプッシュスイッチが押されていない状態で LED0 が点灯、押された状態で LED1 が点灯します。 動作確認が終わったら電源スイッチ 2 つを OFF にします。 LED0 のみしか点灯しない場合は、プッシュスイッチまでの回路が半田付け不良、LED1 が付きっぱなしの場合は半田ブリッジなどが考えられます。目視チェック、テスタなどで原因を突き止めてください。

4.4 サーボの動作確認

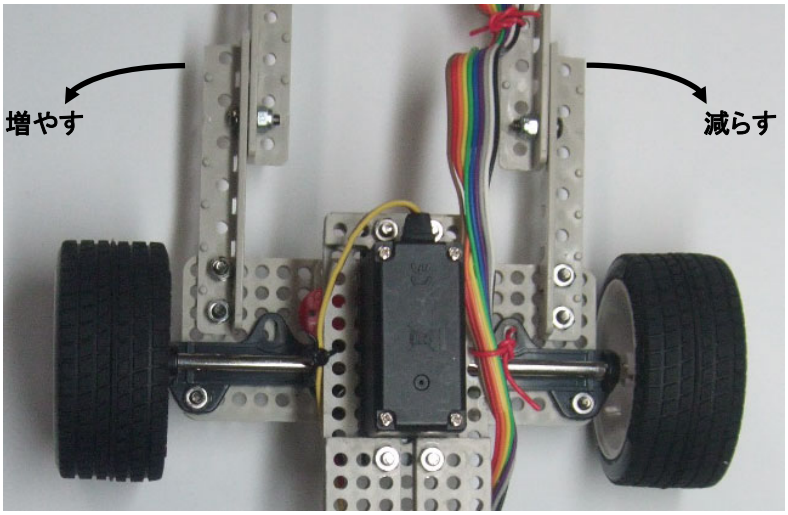
1		モータドライブ基板に接続したサーボが動作するか確認します。フロント部分が動きますので、マイコンカーの回りには何も置かないでください。 ディップスイッチを"0010"の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。
2		サーボが 1 秒毎に、「0 度→右 30 度→左 30 度」の動作を繰り返します。 動作確認が終わったら電源スイッチ 2 つを OFF にします。 サーボが動作しない場合は、サーボまでの回路が半田付け不良、サーボコネクタの向きが逆などが考えられます。また、モータドライブ基板の電源 LED が点灯しているかも確認してください。目視チェック、テスタなどで原因を突き止めてください。
3		キット付属ではないサーボに交換したとき、「0 度→**左** 30 度→**右** 30 度」の動作になる場合、左右の回転が逆なサーボです。その場合、「kit07test_38a.c」の handle 関数内の 425 行の「－(マイナス)」を「＋(プラス)」に変更すれば左右が入れ代わり、動作が「0 度→右 30 度→左 30 度」となります。

※サーボのセンタ調整

マイコンカーの電源を入れたとき、サーボは 0 度になっていなければいけません。しかし、ほとんどの場合は電源を ON にしても 0 度になっていないと思います。これは、サーボのセンタ値(まっすぐにする値)が、サーボによってそれぞれ違うのでマイコンカー1 台 1 台で調整する必要があるためです。「kit07test_38a.c」の SERVO_CENTER の「**3750**」という値を変えてまっすぐになるように調整します。この数値は、26 で約 1 度です。

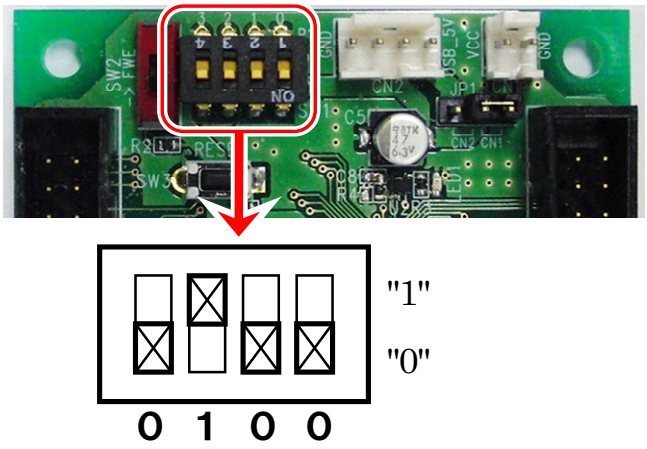
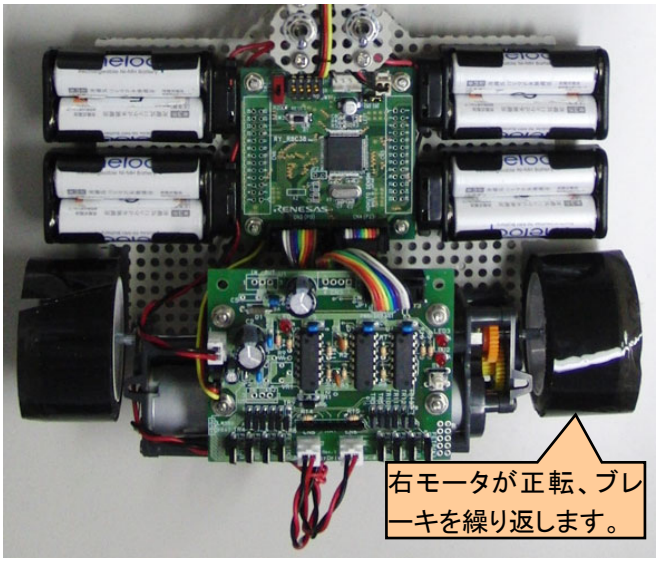
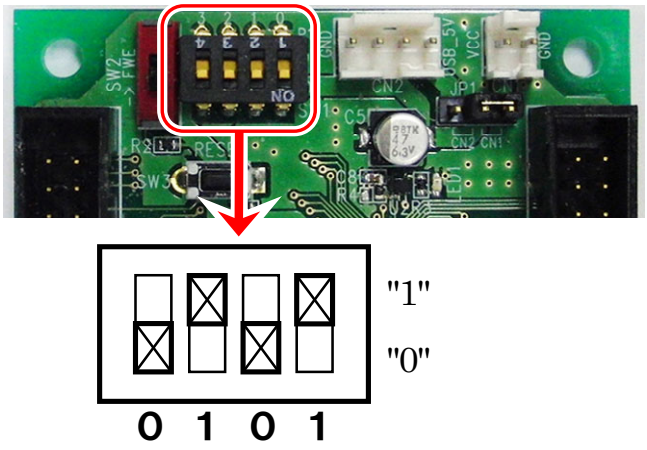
```
46 : #define SERVO_CENTER     3750            /* サーボのセンタ値              */
```

値を増やせば進行方向に向かって左側、減らせば右側に向きます。

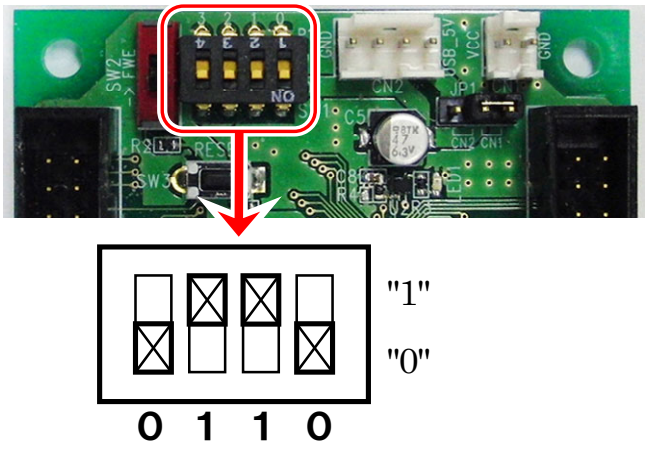
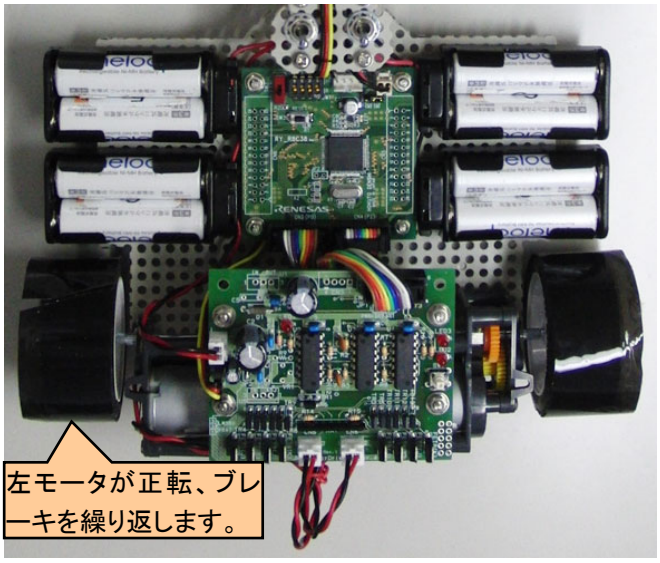
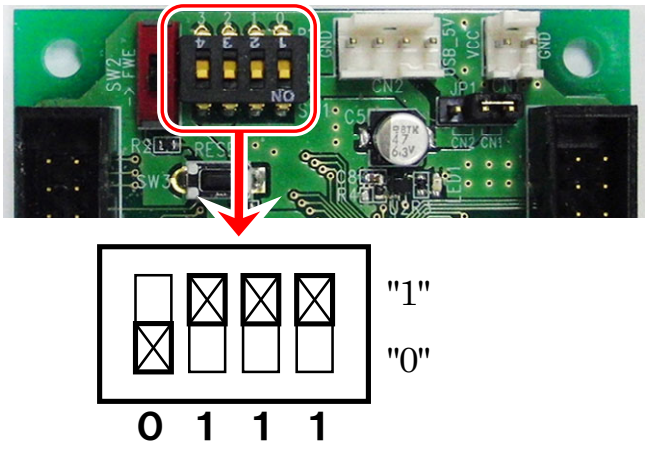


サーボセンタ値の調整は、マイコンカーキット Ver.5 kit07_38a プログラム解説マニュアル(R8C/38A 版)の「7. サーボセンタと最大切れ角の調整」を参照しながら行うと便利です。

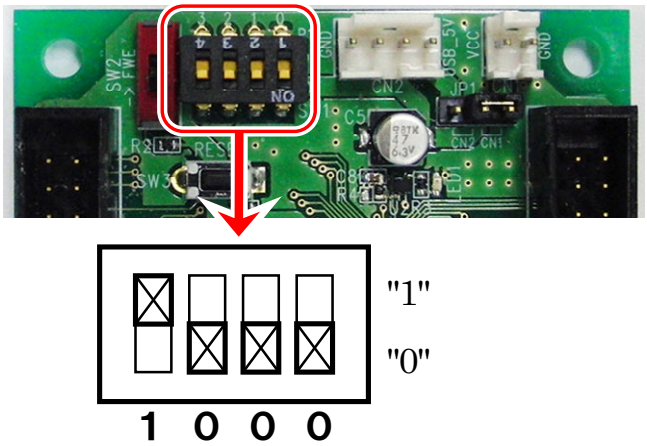
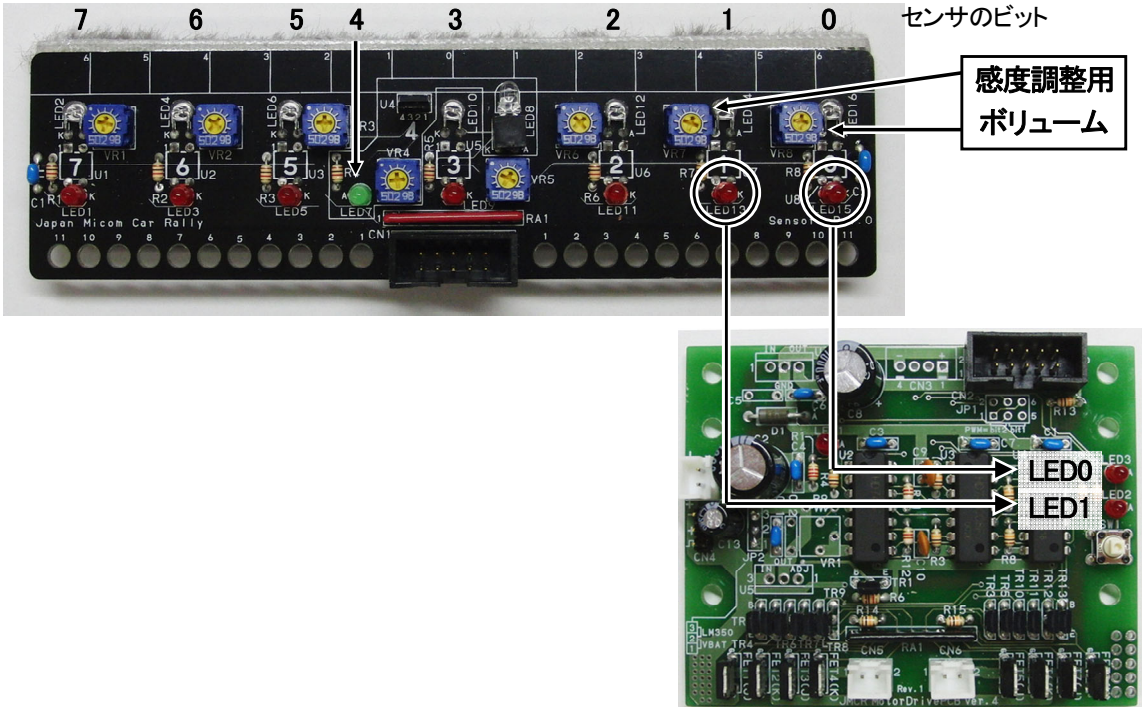
4.5 右モータの動作確認

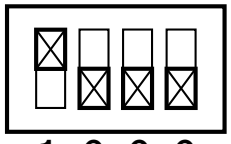
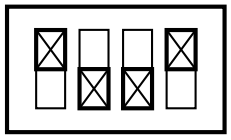
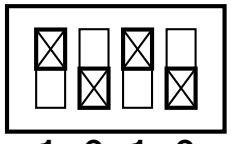
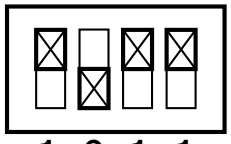
1		右モータが正転、逆転、停止するか確認します。右モータが回りますので、タイヤを持ち上げた状態にしてください。 ディップスイッチを"0100"の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。
2		右モータが 1 秒ごとに「正転→ブレーキ」を繰り返します。 動作確認が終わったら電源スイッチ 2 つを OFF にします。 右モータが正転しない場合は、右モータ制御回路の半田付け不良が考えられます。回転し続ける場合は、半田ブリッジしている可能性があります。目視チェック、テスタなどで原因を突き止めてください。 また、タイヤが逆転した場合は、モータのケーブルが逆です。コネクタの 1 ピンと 2 ピンの端子を入れ替えてください。
3		ディップスイッチを"0101"の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。 右モータが 1 秒ごとに「逆転→ブレーキ」を繰り返します。 動作確認が終わったら電源スイッチ 2 つを OFF にします。 右モータが逆転しない場合は、半田付け不良やショートが考えられます。目視チェック、テスタなどで原因を突き止めてください。

4.6 左モータの動作確認

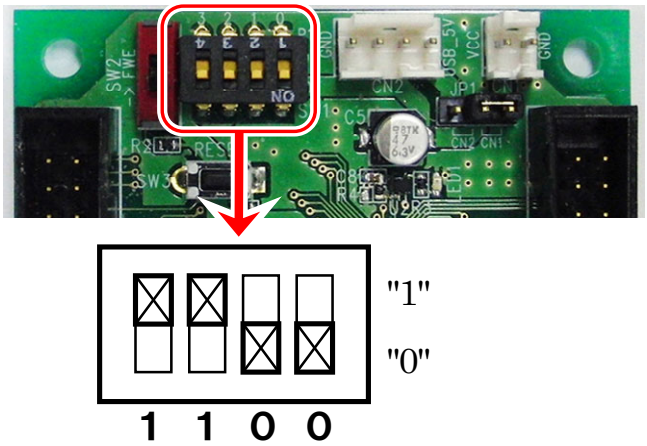
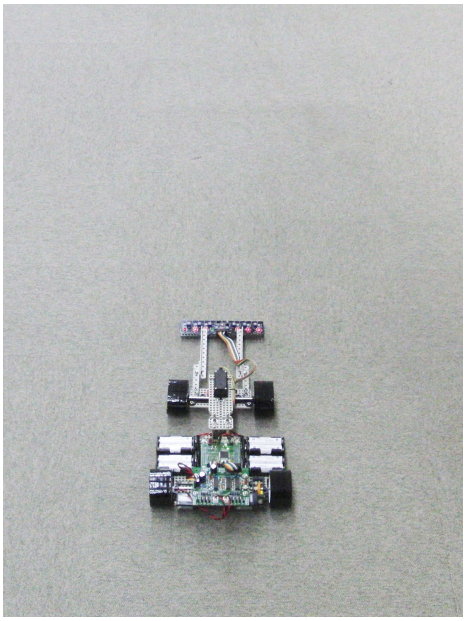
1		左モータが正転、逆転、停止するか確認します。左モータが回りますので、タイヤを持ち上げた状態にしてください。 ディップスイッチを"0110"の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。
2		左モータが 1 秒ごとに「正転→ブレーキ」を繰り返します。 動作確認が終わったら電源スイッチ 2 つを OFF にします。 左モータが正転しない場合は、左モータ制御回路の半田付け不良が考えられます。回転し続ける場合は、半田ブリッジしている可能性があります。目視チェック、テスタなどで原因を突き止めてください。 また、タイヤが逆転した場合は、モータのケーブルが逆です。コネクタの 1 ピンと 2 ピンの端子を入れ替えてください。
3		ディップスイッチを"0111"の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。 左モータが 1 秒ごとに「逆転→ブレーキ」を繰り返します。 動作確認が終わったら電源スイッチ 2 つを OFF にします。 左モータが逆転しない場合は、半田付け不良やショートが考えられます。目視チェック、テスタなどで原因を突き止めてください。

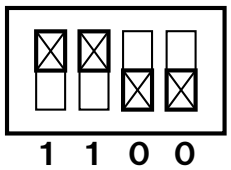
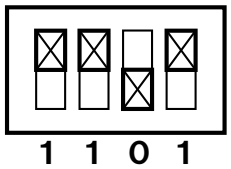
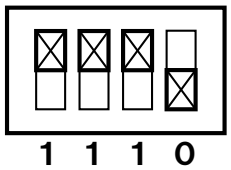
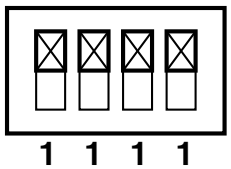
4.7 センサ基板の動作確認

1		ディップスイッチを”1000”の状態で、マイコンカーの電源スイッチを 2 つとも ON にします。
2	 上写真のように、センサ基板のセンサ 0 とセンサ 1 の状態が、モータドライブ基板の 2 つの LED に出力されます。センサ基板上の LED も点灯しますので、同じ反応か確認します。センサの感度はボリュームで調整します。 もしセンサ基板上の LED が点灯しない場合は、センサ基板の半田付け不良、半田ブリッジ、部品の逆差しなどが考えられます。センサ基板上の LED は点灯するのにモータドライブ基板上の LED が点灯しない場合は、コネクタ周りで不具合が発生している可能性があります。目視チェック、テスタなどで原因を突き止めてください。	

3		ディップスイッチ	モータドライブ基板の LED1 に出力される センサ番号	モータドライブ基板の LED0 に出力される センサ番号	同じように左の表のようにディップスイッチを切り替えて、センサ基板のセンサ 7～0 の合計 8 個を確認します。
			1	0	
			3	2	
			5	4 (スタートバー検出)	
			7	6	

4.8 直進性の確認

1		マイコンカーを走らせ、まっすぐに進むか確認します。廊下など平らで直線の長い場所に、マイコンカーを置いてください(ここでは、マイコンカーのコースには置きません)。 ディップスイッチを"1100"の状態にします。
2		マイコンカーの電源スイッチを 2 つとも ON にします。電源をONにしてから、2秒後にPWM値 50%で 2 秒間直進します。廊下など平らで直線の長い場所でマイコンカーを走らせ、まっすぐに進むか確認してください。 曲がってしまう場合は、「**4.4 サーボの動作確認**」で説明したとおり、SERVO_CENTERの値を調整して直進するように調整します。直進性はマイコンカーのスピードが上がると非常に重要になりますので必ずまっすぐ進むように調整してください。最終的には、SERVO_CENTER値を1ずつ調整するぐらいの心がけで調整してください。

<table>
<tr>
<td>3</td>
<td>
<table>
<tr><th>ディップスイッチ</th><th>PWM 値</th><th>停止するまでの時間</th></tr>
<tr><td> 
</td><td>50%</td><td>2 秒</td></tr>
<tr><td> 
</td><td>50%</td><td>5 秒</td></tr>
<tr><td> 
</td><td>100%</td><td>2 秒</td></tr>
<tr><td> 
</td><td>100%</td><td>5 秒</td></tr>
</table>
</td>
<td>直進性の確認は、PWM 値と停止するまでの時間の違いで 4 パターンあります。PWM 値が大きいほど、また停止するまでの時間が長いほど、長く走ります。確保できる長さに応じて 4 つのどれかから選び、廊下など平らで直線の長い場所でマイコンカーを走らせ、まっすぐに進むか確認してください。

※サーボのセンタ調整は、マイコンカーキット Ver.5 kit07_38a プログラム解説マニュアル(R8C/38A 版)の「7. サーボセンタと最大切れ角の調整」で行うと便利です。</td>
</tr>
</table>

4.9 動作確認終了

全機能が正常に動作したなら、走行プログラムを書き込みコースを走らせてみましょう。その前に、調整項目があります。

- マイコンカーキット Ver.5 kit07_38a プログラム解説マニュアル(R8C/38A 版)の「7. サーボセンタと最大切れ角の調整」でサーボセンタと最大切れ角を調整してください。
- マイコンカーキット Ver.5 kit07_38a プログラム解説マニュアル(R8C/38A 版)を参照しながら、ワークスペース「kit07_38a」のプロジェクト「kit07_38a」のプログラムを RY_R8C38 ボードに書き込み、コースを走らせてみてください。

5. プログラムソース

5.1 「kit07test_38a.c」のプログラム内容

```
 1 : /************************************************************************/
 2 : /* 対象マイコン R8C/38A                                                 */
 3 : /* ﾌｧｲﾙ内容     マイコンカーキット  動作確認プログラム(R8C/38A版)        */
 4 : /* バージョン   Ver.1.01                                                */
 5 : /* Date         2011.03.14                                              */
 6 : /* Copyright    ジャパンマイコンカーラリー実行委員会                    */
 7 : /************************************************************************/
 8 :
 9 : /*
10 : マイコンカーキット用センサ基板Ver.4.1、モータドライブ基板Ver.4の
11 : 動作確認を行います。
12 : マイコンボードのディップスイッチにより動作確認する内容を変更します。
13 :    DipSW
14 : bit3 2 1 0
15 :    0 0 0 0 LEDの確認          LEDが0.5秒間隔で交互に点灯
16 :    0 0 0 1 ﾌﾟｯｼｭｽｲｯﾁの確認   OFF時：LED0点灯 ON時：LED1点灯
17 :    0 0 1 0 サーボの確認       0°→右30°→左30° の繰り返し
18 :    0 0 1 1 動作無し
19 :    0 1 0 0 右モータの確認     正転→ブレーキの繰り返し
20 :    0 1 0 1                    逆転→ブレーキの繰り返し
21 :    0 1 1 0 左モータの確認     正転→ブレーキの繰り返し
22 :    0 1 1 1                    逆転→ブレーキの繰り返し
23 :
24 :    1 0 0 0 センサ確認         センサbit1,0をLED1,0に出力
25 :    1 0 0 1                    センサbit3,2をLED1,0に出力
26 :    1 0 1 0                    センサbit5,4をLED1,0に出力
27 :    1 0 1 1                    センサbit7,6をLED1,0に出力
28 :
29 :    1 1 0 0 直進性の確認       PWM  50%で前進、 2秒後ストップ
30 :    1 1 0 1 直進性の確認       PWM  50%で前進、 5秒後ストップ
31 :    1 1 1 0 直進性の確認       PWM 100%で前進、 2秒後ストップ
32 :    1 1 1 1 直進性の確認       PWM 100%で前進、 5秒後ストップ
33 : */
34 :
35 : /*====================================*/
36 : /* インクルード                       */
37 : /*====================================*/
38 : #include "sfr_r838a.h"                 /* R8C/38A SFRの定義ファイル    */
39 :
40 : /*====================================*/
41 : /* シンボル定義                       */
42 : /*====================================*/
43 :
44 : /* 定数設定 */
45 : #define PWM_CYCLE       39999         /* モータPWMの周期              */
46 : #define SERVO_CENTER    3750          /* サーボのセンタ値             */
47 : #define HANDLE_STEP     22            /* 1° 分の値                   */
48 :
49 : /*====================================*/
50 : /* プロトタイプ宣言                   */
51 : /*====================================*/
52 : void init( void );
53 : unsigned char sensor_inp_all( unsigned char mask );
54 : unsigned char dipsw_get( void );
55 : unsigned char pushsw_get( void );
56 : void led_out( unsigned char led );
57 : void motor( int accele_l, int accele_r );
58 : void handle( int angle );
59 :
60 : /*====================================*/
61 : /* グローバル変数の宣言               */
62 : /*====================================*/
63 : unsigned long   cnt0;                  /* timer関数用                  */
64 : unsigned long   cnt1;                  /* main内で使用                 */
65 :
66 : /************************************************************************/
67 : /* メインプログラム                                                     */
68 : /************************************************************************/
69 : void main( void )
70 : {
71 :     unsigned char   now_sw;           /* 現在ディップスイッチ記憶     */
72 :     unsigned char   before_sw;        /* 前回ディップスイッチ記憶     */
73 :     unsigned char   c;                /* 作業用                       */
74 :     int             i;                /* 作業用                       */
75 :
```

```
 76 :      /* マイコン機能の初期化 */
 77 :      init();                             /* 初期化                          */
 78 :      asm(" fset I ");                    /* 全体の割り込み許可              */
 79 :
 80 :      /* 変数初期化 */
 81 :      before_sw = dipsw_get();
 82 :      cnt1 = 0;
 83 :
 84 :      /* マイコンカーの状態初期化 */
 85 :      handle( 0 );
 86 :      motor( 0, 0 );
 87 :      led_out( 0x0 );
 88 :
 89 :      while( 1 ) {
 90 :      /* ディップスイッチ読み込み */
 91 :      now_sw = dipsw_get();
 92 :
 93 :      /* 前回のスイッチ値と比較 */
 94 :      if( before_sw != now_sw ) {
 95 :          /* 不一致なら前回値更新、タイマ値のクリア */
 96 :          before_sw = now_sw;
 97 :          cnt1 = 0;
 98 :      }
 99 :
100 :      /* ディップスイッチの値により動作確認モードの選択 */
101 :      switch( now_sw ) {
102 :
103 :          /* LEDの確認 LEDが0.5秒間隔で交互に点灯 */
104 :          case 0:
105 :              if( cnt1 < 500 ) {
106 :                  led_out( 0x1 );
107 :              } else if( cnt1 < 1000 ) {
108 :                  led_out( 0x2 );
109 :              } else {
110 :                  cnt1 = 0;
111 :              }
112 :              break;
113 :
114 :          /* プッシュスイッチの確認 OFF時：LED0点灯 ON時：LED1点灯 */
115 :          case 1:
116 :              led_out( pushsw_get() + 1 );
117 :              break;
118 :
119 :          /* サーボの確認 0° →右30° →左30° の繰り返し */
120 :          case 2:
121 :              if( cnt1 < 1000 ) {
122 :                  handle( 0 );
123 :              } else if( cnt1 < 2000 ) {
124 :                  handle( 30 );
125 :              } else if( cnt1 < 3000 ) {
126 :                  handle( -30 );
127 :              } else {
128 :                  cnt1 = 0;
129 :              }
130 :              break;
131 :
132 :          /* 何もしない */
133 :          case 3:
134 :              break;
135 :
136 :          /* 右モータの確認 正転→ブレーキの繰り返し */
137 :          case 4:
138 :              if( cnt1 < 1000 ) {
139 :                  motor( 0, 100 );
140 :              } else if( cnt1 < 2000 ) {
141 :                  motor( 0, 0 );
142 :              } else {
143 :                  cnt1 = 0;
144 :              }
145 :              break;
146 :
147 :          /* 右モータの確認 逆転→ブレーキの繰り返し */
148 :          case 5:
149 :              if( cnt1 < 1000 ) {
150 :                  motor( 0, -100 );
151 :              } else if( cnt1 < 2000 ) {
152 :                  motor( 0, 0 );
153 :              } else {
154 :                  cnt1 = 0;
155 :              }
156 :              break;
157 :
158 :          /* 左モータの確認 正転→ブレーキの繰り返し */
159 :          case 6:
160 :              if( cnt1 < 1000 ) {
161 :                  motor( 100, 0 );
162 :              } else if( cnt1 < 2000 ) {
163 :                  motor( 0, 0 );
164 :              } else {
165 :                  cnt1 = 0;
```

```
166 :             }
167 :             break;
168 :
169 :         /* 左モータの確認 逆転→ブレーキの繰り返し */
170 :         case 7:
171 :             if( cnt1 < 1000 ) {
172 :                 motor( -100, 0 );
173 :             } else if( cnt1 < 2000 ) {
174 :                 motor( 0, 0 );
175 :             } else {
176 :                 cnt1 = 0;
177 :             }
178 :             break;
179 :
180 :         /* センサ基板の確認 センサbit1,0をLED1,0に出力 */
181 :         case 8:
182 :             c = sensor_inp_all( 0x03 );
183 :             led_out( c );
184 :             break;
185 :
186 :         /* センサ基板の確認 センサbit3,2をLED1,0に出力 */
187 :         case 9:
188 :             c = sensor_inp_all( 0x0c );
189 :             c = c >> 2;
190 :             led_out( c );
191 :             break;
192 :
193 :         /* センサ基板の確認 センサbit5,4をLED1,0に出力 */
194 :         case 10:
195 :             c = sensor_inp_all( 0x30 );
196 :             c = c >> 4;
197 :             led_out( c );
198 :             break;
199 :
200 :         /* センサ基板の確認 センサbit7,6をLED1,0に出力 */
201 :         case 11:
202 :             c = sensor_inp_all( 0xc0 );
203 :             c = c >> 6;
204 :             led_out( c );
205 :             break;
206 :
207 :         /* 直進性の確認 PWM  50%で前進、 2秒後ストップ */
208 :         case 12:
209 :             if( cnt1 < 2000 ) {
210 :                 motor( 0, 0 );
211 :             } else if( cnt1 < 4000 ) {
212 :                 motor( 50, 50 );
213 :             } else {
214 :                 motor( 0, 0 );
215 :             }
216 :             break;
217 :
218 :         /* 直進性の確認 PWM  50%で前進、 5秒後ストップ */
219 :         case 13:
220 :             if( cnt1 < 2000 ) {
221 :                 motor( 0, 0 );
222 :             } else if( cnt1 < 7000 ) {
223 :                 motor( 50, 50 );
224 :             } else {
225 :                 motor( 0, 0 );
226 :             }
227 :             break;
228 :
229 :         /* 直進性の確認 PWM 100%で前進、 2秒後ストップ */
230 :         case 14:
231 :             if( cnt1 < 2000 ) {
232 :                 motor( 0, 0 );
233 :             } else if( cnt1 < 4000 ) {
234 :                 motor( 100, 100 );
235 :             } else {
236 :                 motor( 0, 0 );
237 :             }
238 :             break;
239 :
240 :         /* 直進性の確認 PWM 100%で前進、 5秒後ストップ */
241 :         case 15:
242 :             if( cnt1 < 2000 ) {
243 :                 motor( 0, 0 );
244 :             } else if( cnt1 < 7000 ) {
245 :                 motor( 100, 100 );
246 :             } else {
247 :                 motor( 0, 0 );
248 :             }
249 :             break;
250 :
251 :         /* どれでもないなら */
252 :         default:
253 :             break;
254 :     }
255 :     }
```

```
256 :  }
257 :
258 :  /************************************************************************/
259 :  /* R8C/38A スペシャルファンクションレジスタ(SFR)の初期化                */
260 :  /************************************************************************/
261 :  void init( void )
262 :  {
263 :      int i;
264 :
265 :      /* クロックをXINクロック(20MHz)に変更 */
266 :      prc0  = 1;                          /* プロテクト解除               */
267 :      cm13  = 1;                          /* P4_6,P4_7をXIN-XOUT端子にする*/
268 :      cm05  = 0;                          /* XINクロック発振              */
269 :      for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約10ms) */
270 :      ocd2  = 0;                          /* システムクロックをXINにする  */
271 :      prc0  = 0;                          /* プロテクトON                 */
272 :
273 :      /* ポートの入出力設定 */
274 :      prc2 = 1;                           /* PD0のプロテクト解除          */
275 :      pd0 = 0x00;                         /* 7-0:センサ基板Ver.4.1        */
276 :      pd1 = 0xd0;                         /* 5:RXD0 4:TXD0 3-0:DIP SW     */
277 :      pd2 = 0xfe;                         /* 7-0:モータドライブ基板Ver.4  */
278 :      pd3 = 0xff;                         /*                              */
279 :      p4  = 0x20;                         /* P4_5のLED:初期は点灯         */
280 :      pd4 = 0xb8;                         /* 7:XOUT 6:XIN 5:LED 2:VREF    */
281 :      pd5 = 0xff;                         /*                              */
282 :      pd6 = 0xff;                         /*                              */
283 :      pd7 = 0xff;                         /*                              */
284 :      pd8 = 0xff;                         /*                              */
285 :      pd9 = 0x3f;                         /*                              */
286 :      pur0 = 0x04;                        /* P1_3～P1_0のプルアップON     */
287 :
288 :      /* タイマRBの設定 */
289 :      /* 割り込み周期 = 1 / 20[MHz]   * (TRBPRE+1) * (TRBPR+1)
290 :                      = 1 / (20*10^6) * 200        * 100
291 :                      = 0.001[s] = 1[ms]
292 :      */
293 :      trbmr  = 0x00;                      /* 動作モード、分周比設定       */
294 :      trbpre = 200-1;                     /* プリスケーラレジスタ         */
295 :      trbpr  = 100-1;                     /* プライマリレジスタ           */
296 :      trbic  = 0x07;                      /* 割り込み優先レベル設定       */
297 :      trbcr  = 0x01;                      /* カウント開始                 */
298 :
299 :      /* タイマRD リセット同期PWMモードの設定*/
300 :      /* PWM周期 = 1 / 20[MHz]   * カウントソース * (TRDGRA0+1)
301 :                 = 1 / (20*10^6) * 8              * 40000
302 :                 = 0.016[s] = 16[ms]
303 :      */
304 :      trdpsr0 = 0x08;                     /* TRDIOB0,C0,D0端子設定        */
305 :      trdpsr1 = 0x05;                     /* TRDIOA1,B1,C1,D1端子設定     */
306 :      trdmr   = 0xf0;                     /* バッファレジスタ設定         */
307 :      trdfcr  = 0x01;                     /* リセット同期PWMモードに設定  */
308 :      trdcr0  = 0x23;                     /* ソースカウントの選択:f8      */
309 :      trdgra0 = trdgrc0 = PWM_CYCLE;      /* 周期                         */
310 :      trdgrb0 = trdgrd0 = 0;              /* P2_2端子のON幅設定           */
311 :      trdgra1 = trdgrc1 = 0;              /* P2_4端子のON幅設定           */
312 :      trdgrb1 = trdgrd1 = SERVO_CENTER;   /* P2_5端子のON幅設定           */
313 :      trdoer1 = 0xcd;                     /* 出力端子の選択               */
314 :      trdstr  = 0x0d;                     /* TRD0カウント開始             */
315 :  }
316 :
317 :  /************************************************************************/
318 :  /* タイマRB 割り込み処理                                                */
319 :  /************************************************************************/
320 :  #pragma interrupt intTRB(vect=24)
321 :  void intTRB( void )
322 :  {
323 :      cnt0++;
324 :      cnt1++;
325 :  }
326 :
327 :  /************************************************************************/
328 :  /* センサ状態検出(スタートバーセンサを含めすべてのセンサ)               */
329 :  /* 引数   マスク値                                                      */
330 :  /* 戻り値 センサ値                                                      */
331 :  /************************************************************************/
332 :  unsigned char sensor_inp_all( unsigned char mask )
333 :  {
334 :      unsigned char sensor;
335 :
336 :      sensor  = ~p0;
337 :      sensor &= mask;
338 :
339 :      return sensor;
340 :  }
341 :
```

```
342 :  /************************************************************************/
343 :  /* ディップスイッチ値読み込み                                           */
344 :  /* 戻り値 スイッチ値 0～15                                              */
345 :  /************************************************************************/
346 :  unsigned char dipsw_get( void )
347 :  {
348 :      unsigned char sw;
349 :
350 :      sw = p1 & 0x0f;                     /* P1_3～P1_0読み込み           */
351 :
352 :      return  sw;
353 :  }
354 :
355 :  /************************************************************************/
356 :  /* プッシュスイッチ値読み込み                                           */
357 :  /* 戻り値 スイッチ値 ON:1 OFF:0                                         */
358 :  /************************************************************************/
359 :  unsigned char pushsw_get( void )
360 :  {
361 :      unsigned char sw;
362 :
363 :      sw  = ~p2;                          /* スイッチのあるポート読み込み */
364 :      sw &= 0x01;
365 :
366 :      return  sw;
367 :  }
368 :
369 :  /************************************************************************/
370 :  /* LED制御                                                              */
371 :  /* 引数  スイッチ値 LED0:bit0 LED1:bit1  "0":消灯 "1":点灯              */
372 :  /* 例)0x3→LED1:ON LED0:ON  0x2→LED1:ON LED0:OFF                        */
373 :  /************************************************************************/
374 :  void led_out( unsigned char led )
375 :  {
376 :      unsigned char data;
377 :
378 :      led = ~led;
379 :      led <<= 6;
380 :      data = p2 & 0x3f;
381 :      p2 = data | led;
382 :  }
383 :
384 :  /************************************************************************/
385 :  /* モータ速度制御                                                       */
386 :  /* 引数   左モータ:-100～100、右モータ:-100～100                        */
387 :  /*        0で停止、100で正転100%、-100で逆転100%                        */
388 :  /* 戻り値 なし                                                          */
389 :  /************************************************************************/
390 :  void motor( int data1, int data2 )
391 :  {
392 :      int    motor_r, motor_l, sw_data;
393 :
394 :      sw_data = dipsw_get() + 5;
395 :      motor_l = data1 * sw_data / 20;
396 :      motor_r = data2 * sw_data / 20;
397 :
398 :      /* 左モータ制御 */
399 :      if( motor_l >= 0 ) {
400 :          p2 &= 0xfd;
401 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
402 :      } else {
403 :          p2 |= 0x02;
404 :          trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
405 :      }
406 :
407 :      /* 右モータ制御 */
408 :      if( motor_r >= 0 ) {
409 :          p2 &= 0xf7;
410 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
411 :      } else {
412 :          p2 |= 0x08;
413 :          trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
414 :      }
415 :  }
416 :
417 :  /************************************************************************/
418 :  /* サーボハンドル操作                                                   */
419 :  /* 引数   サーボ操作角度：-90～90                                       */
420 :  /*        -90で左へ90度、0でまっすぐ、90で右へ90度回転                   */
421 :  /************************************************************************/
422 :  void handle( int angle )
423 :  {
424 :      /* サーボが左右逆に動く場合は、「-」を「+」に替えてください */
425 :      trdgrd1 = SERVO_CENTER - angle * HANDLE_STEP;
426 :  }
427 :
428 :  /************************************************************************/
429 :  /* end of file                                                          */
430 :  /************************************************************************/
```

5.2 「startup.c」のプログラム内容

```
 1 :  /************************************************************************/
 2 :  /* 対象マイコン R8C/38A                                                 */
 3 :  /* ﾌｧｲﾙ内容     スタートアッププログラム(C言語版)                       */
 4 :  /* バージョン   Ver.1.22                                                */
 5 :  /* Date         2012.03.08                                              */
 6 :  /* Copyright    ジャパンマイコンカーラリー実行委員会                    */
 7 :  /*              ルネサスマイコンカーラリー事務局                        */
 8 :  /*              日立インターメディックス株式会社                        */
 9 :  /************************************************************************/
10 :  /*====================================*/
11 :  /* プロトタイプ宣言(ローカル)         */
12 :  /*====================================*/
13 :  void main( void );
14 :
15 :  /*====================================*/
16 :  /* スタックサイズの設定               */
17 :  /*====================================*/
18 :  #pragma STACKSIZE       0x100
19 :  #pragma ISTACKSIZE      0x100
20 :
21 :  /*====================================*/
22 :  /* CPUレジスタの宣言                  */
23 :  /*====================================*/
24 :  #pragma CREG    _flg_   flg
25 :  #pragma CREG    _isp_   isp
26 :  #pragma CREG    _sp_    sp
27 :  #pragma CREG    _sb_    sb
28 :  #pragma CREG    _fb_    fb
29 :  #pragma CREG    _intbh_ intbh
30 :  #pragma CREG    _intbl_ intbl
31 :
32 :  unsigned int _flg_;
33 :  unsigned int _sb_;
34 :  unsigned int _fb_;
35 :  unsigned int *_sp_;
36 :  unsigned int *_isp_;
37 :  unsigned int *_intbh_;
38 :  unsigned int *_intbl_;
39 :
40 :  /*====================================*/
41 :  /* SBの値をコンパイラに設定           */
42 :  /*====================================*/
43 :      _asm("       .glb    __SB__¥n"
44 :           "__SB__ .equ    0400H   ");
45 :
46 :  /*====================================*/
47 :  /* オプション機能選択レジスタの設定   */
48 :  /*====================================*/
49 :      _asm("  .ofsreg 0BFH ");            /* OFS = 0xbf (ﾊﾟﾜｰｵﾝﾘｾｯﾄ使用)  */
50 :
51 :  /*====================================*/
52 :  /* IDコードの設定                     */
53 :  /*====================================*/
54 :      _asm("  .id ""¥"#FFFFFFFFFFFFFF¥" ");
55 :
56 :  /*====================================*/
57 :  /* RAMを初期化する関数の定義          */
58 :  /*====================================*/
59 :  #define scopy(X,Y,Z)    _asm("  .initsct "X","Y","Z"¥n"¥
60 :                  "   .initsct "X"I,rom"Y",noalign¥n"¥
61 :                  "   mov.w   #(topof "X"I)&0ffffH,A0¥n"¥
62 :                  "   mov.b   #(topof "X"I)>>16,R1H¥n"¥
63 :                  "   mov.w   #(topof "X")&0ffffH,A1¥n"¥
64 :                  "   mov.w   #sizeof "X",R3¥n"¥
65 :                  "   smovf.b")
66 :
67 :  #define sclear(X,Y,Z)   _asm("  .initsct "X","Y","Z"¥n"¥
68 :                  "   mov.b   #00H,R0L¥n"¥
69 :                  "   mov.w   #(topof "X") ,A1¥n"¥
70 :                  "   mov.w   #sizeof "X",R3¥n"¥
71 :                  "   sstr.b")
72 :
73 :  /*====================================*/
74 :  /* セクションの先頭アドレスの型定義   */
75 :  /*====================================*/
76 :  extern unsigned int _stack_top;
77 :  extern unsigned int _istack_top;
78 :  extern unsigned int _vector_top;
79 :
```

```
 80 :  /*====================================*/
 81 :  /* ヒープ領域の設定                   */
 82 :  /*====================================*/
 83 :  #pragma sectaddress heap_NE,DATA
 84 :  #define __HEAPSIZE__    0x100
 85 :
 86 :  unsigned char heap_area[ __HEAPSIZE__ ]; /* ヒープ領域確保             */
 87 :  extern unsigned char  _far *_mnext;     /* 次に使用できるﾒﾓﾘの先頭ｱﾄﾞﾚｽ */
 88 :  extern unsigned long        _msize;     /* 残りのバイト数             */
 89 :
 90 :  /*====================================*/
 91 :  /* 固定割り込みベクタアドレスの設定   */
 92 :  /*====================================*/
 93 :  #pragma sectaddress fvector,ROMDATA 0xffd8
 94 :
 95 :      _asm("  .addr 0FFFFFFFH");          /* 予約                       */
 96 :      _asm("  .byte 0FFH");               /* OFS2                       */
 97 :  #pragma interrupt/v _dummy_int          /* 未定義命令割り込みベクタ   */
 98 :  #pragma interrupt/v _dummy_int          /* オーバフロー割り込みベクタ */
 99 :  #pragma interrupt/v _dummy_int          /* BRK命令割り込みベクタ      */
100 :  #pragma interrupt/v _dummy_int          /* アドレス一致割り込みベクタ */
101 :  #pragma interrupt/v _dummy_int          /* ｼﾝｸﾞﾙｽﾃｯﾌﾟ割り込みベクタ   */
102 :  #pragma interrupt/v _dummy_int          /* ｳｫｯﾁﾄﾞｯｸﾞﾀｲﾏなどの割り込みﾍﾞｸﾀ*/
103 :  #pragma interrupt/v _dummy_int          /* ｱﾄﾞﾚｽﾌﾞﾚｲｸ割り込みベクタ   */
104 :  #pragma interrupt/v _dummy_int          /* 予約                       */
105 :  #pragma interrupt/v start               /* リセットベクタ             */
106 :
107 :  /*====================================*/
108 :  /* 固定割り込みプログラム             */
109 :  /*====================================*/
110 :  #pragma sectaddress interrupt,CODE
111 :  #pragma interrupt _dummy_int()
112 :  void _dummy_int(void)
113 :  {
114 :      /* ダミー関数 */
115 :  }
116 :
117 :  /*====================================*/
118 :  /* 可変割り込みベクタの設定           */
119 :  /*====================================*/
120 :  #pragma sectaddress vector,ROMDATA
121 :
122 :      /* ここではセクション名の設定のみ行う */
123 :
124 :  /*====================================*/
125 :  /* RAMの初期化                        */
126 :  /*====================================*/
127 :  #pragma sectaddress program,CODE
128 :  void initsct(void)
129 :  {
130 :      sclear("bss_SE","data","align");
131 :      sclear("bss_SO","data","noalign");
132 :      sclear("bss_NE","data","align");
133 :      sclear("bss_NO","data","noalign");
134 :
135 :      scopy("data_SE","data","align");
136 :      scopy("data_SO","data","noalign");
137 :      scopy("data_NE","data","align");
138 :      scopy("data_NO","data","noalign");
139 :  }
140 :
141 :  /*====================================*/
142 :  /* スタートアッププログラム           */
143 :  /*====================================*/
144 :  #pragma entry start
145 :  void start( void )
146 :  {
147 :      _isp_   = &_istack_top;             /* ISPに割り込みｽﾀｯｸのｱﾄﾞﾚｽ設定 */
148 :      _flg_   = 0x0080;                   /* FLGのU="1"                 */
149 :      _sp_    = &_stack_top;              /* USPにﾕｰｻﾞｽﾀｯｸのｱﾄﾞﾚｽ設定   */
150 :      _sb_    = 0x0400U;                  /* SB相対アドレッシングの設定 */
151 :      _intbh_ = (unsigned int*)0x00;      /* INTBH = vector(上位)に設定 */
152 :      _intbl_ = &_vector_top;             /* INTBL = vector(下位)に設定 */
153 :      initsct();                          /* RAMの初期化                */
154 :      _mnext = &heap_area[0];             /* ヒープ領域変数の設定       */
155 :      _msize = (unsigned long)__HEAPSIZE__;
156 :      _fb_    = 0U;                       /* FB = 0                     */
157 :
158 :      main();                             /* main関数実行               */
159 :
160 :      while(1);
161 :  }
162 :
163 :  /************************************************************************/
164 :  /* end of file                                                          */
165 :  /************************************************************************/
```