

ミニマイコンカー製作キット Ver.2 拡張用オプション

基板マイコンカー 補足マニュアル

第 1.02 版

2015.09.22

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、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いたします。
- ・すべての商標および登録商標は、それぞれの所有者に帰属します。

連絡先

株式会社 日立ドキュメントソリューションズ

〒135-0016 東京都江東区東陽六丁目 3 番 2 号 イースト 21 タワー

E-mail:himdx.m-carrally.dd@hitachi.com

目次

1. 概要.....	1
2. マイコンボードの改造.....	2
2.1 概要.....	2
2.2 改造.....	2
2.3 改造後の USB ケーブル接続手順.....	3
3. 動作確認の手順.....	4
3.1 概要.....	4
3.2 動作確認プログラムの改造.....	4
3.3 動作確認プログラムの書き込み.....	5
3.4 Tera Term を使った動作確認.....	6
4. 液晶の使い方.....	8
4.1 液晶の接続.....	8
4.2 液晶を制御する関数.....	9
4.2.1 液晶の初期化.....	9
4.2.2 液晶表示処理.....	9
4.2.3 液晶表示処理.....	10
4.2.4 液晶表示処理.....	10
5. プッシュスイッチの使い方.....	11
5.1 プッシュスイッチ.....	11
5.2 プッシュスイッチを制御する関数.....	11
5.2.1 プッシュスイッチ関係処理の初期化.....	11
5.2.2 プッシュスイッチ検出処理.....	11
5.2.3 プッシュスイッチの現在情報取得.....	12
5.2.4 プッシュスイッチのフラグ情報取得.....	12

1. 概要

本マニュアルは、「基板マイコンカー製作キット 組み立てマニュアル」で組み立てた基板マイコンカーについての補足説明です。

本マニュアルで説明している内容を下記に示します。

概要	詳細	参照先
マイコンボードの改造	POWER LED が点灯しない、USB ケーブル経由で基板マイコンカーの回路に 5V が供給されて、USB 機器が不安定になってしまうことについて、解説します。	2. マイコンボードの改造
動作確認の手順	TeraTerm などの通信ソフトとうまく接続できないことがあります。うまく接続できる手順を解説します。	3. 動作確認の手順
液晶	液晶を制御する関数について解説します。	4. 液晶の使い方
プッシュスイッチ	プッシュスイッチ4個の状態を検出する関数について解説します。	5. プッシュスイッチの使い方

2. マイコンボードの改造

2.1 概要

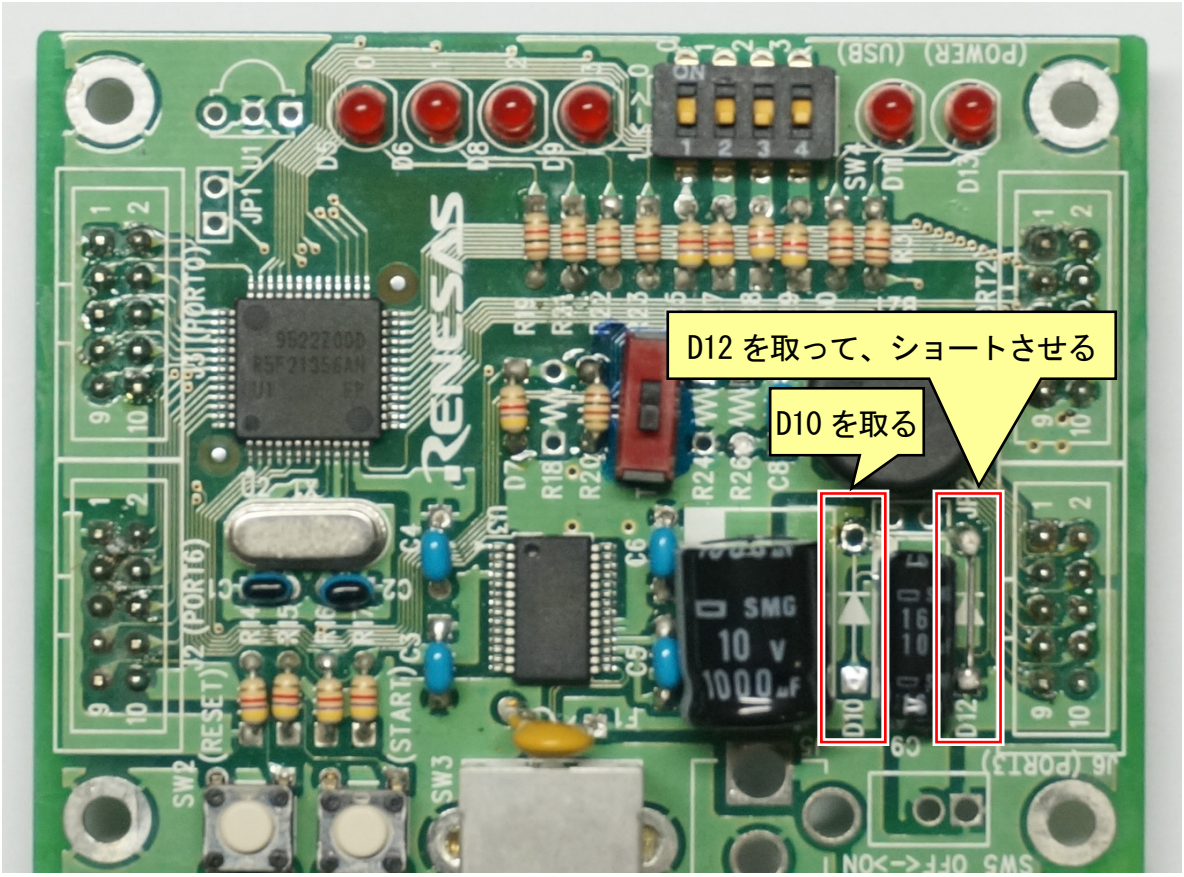
「基板マイコンカー製作キット 組み立てマニュアル」のマイコンボードについて、下記の問い合わせがありました。

- ①POWER LED が点灯しない
- ②USB ケーブル経由で基板マイコンカーの回路に 5V が供給されて、USB 機器が不安定になってしまう

今回、部品を2個取り外し、一部ショートさせることにより、下記の様に対応させます。

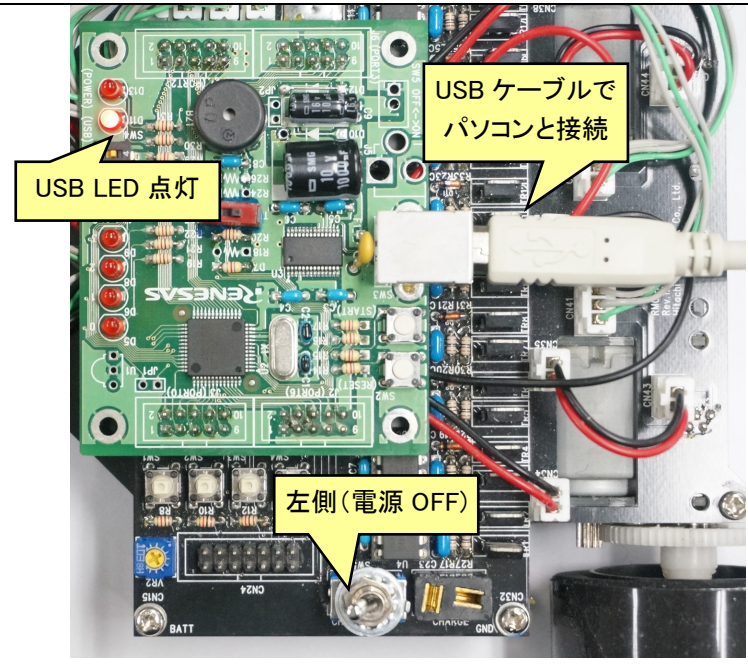
- ①電源を入れると、POWER LED が点灯する
- ②USB ケーブルからは、基板マイコンカーに電源を供給しないようにする

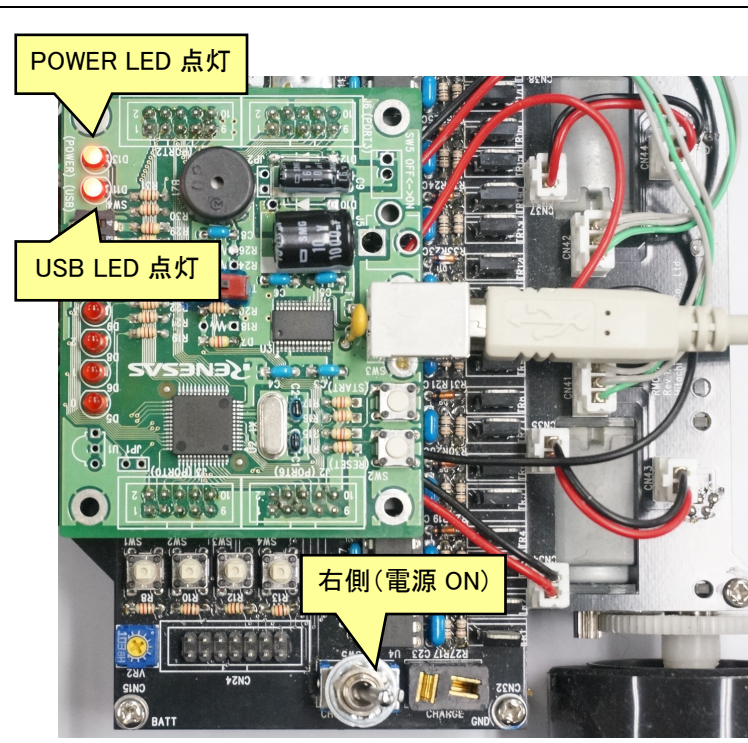
2.2 改造



1	D10 と D12 を取ります。無理にダイオードを半田ごてで取るとランドのスルーホールが取れることがあるので、ニッパーなので切って外すのが簡単です。その後、D12 のランド間を、余ったリード線などでショートさせます。写真は部品面でショートさせていますが、半田面(写真の裏面)でショートさせると簡単に半田付けすることができます。
---	---

2.3 改造後の USB ケーブル接続手順

1		<p>基板マイコンカーの電源スイッチ OFF の状態です。</p> <p>USB ケーブルを接続すると、USB の LED は点灯します。</p> <p>しかし、回路には電源が供給されなくなったのでパソコン側でシリアルポートは認識されません。</p>
---	--	---

2		<p>電源スイッチを ON にすると「POWER LED」が点灯し、基板マイコンカーの回路に電源が供給されます。</p>
---	---	--

3		<p>この状態で、Tera Term など通信ソフトにはシリアルポートが表示されます。</p>
---	--	---

3. 動作確認の手順

3.1 概要

「基板マイコンカー製作キット C 言語走行プログラム解説マニュアル」の内容で動作確認しようとしても、TeraTerm などの通信ソフトと接続できないことがあります。接続できる手順を説明します。

3.2 動作確認プログラムの改造

1		<p>ルネサス統合開発環境で、ワークスペース「mini_mcr2_rmc_frame」を開きます。</p> <p>①プロジェクト「mini_mcr_test」上で右クリックします。</p> <p>②「アクティブプロジェクトに設定」を選択して、プロジェクトを有効にします。</p>
---	--	---

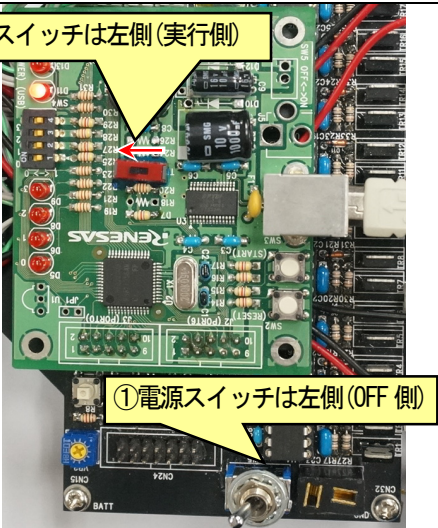
2		<p>「mini_mcr_test.c」でダブルクリックして、動作確認用のプログラムを開きます。</p>
---	--	--

3	<pre> 69 : void main(void) 70 : { 71 : int i = 0; 72 : 73 : // 初期化 74 : init(); 75 : init_sensor(); 76 : 77 : init_uart0_printf(SPEED_9600); 78 : init_lcd(); 79 : init_switch(); 80 : 81 : p6_1 = 0; 82 : while(1) { 83 : if(get_sw_flag(SW_3) != 0) break; // スイッチが押されたらループを抜ける 84 : } </pre>
	<p>81～84行を追加します。「SW_3」(基板のSW4)を押すまで、83行を無限ループするプログラムです。追加できたら、「ビルド→ビルド」を実行して、エラーが0であることを確認してください。</p>

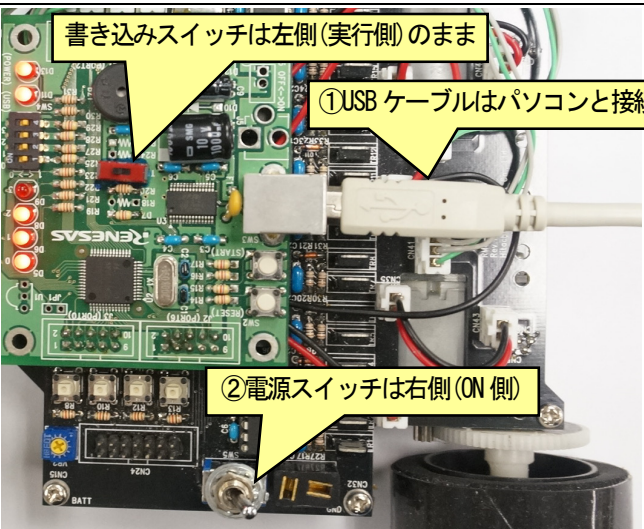
3.3 動作確認プログラムの書き込み

1		<p>①基板マイコンカーの電源をOFFにしておきます。</p> <p>②マイコンボードの書き込みスイッチは右側(書き込み側)にします。</p> <p>③USB ケーブルを接続します。</p>
---	--	---

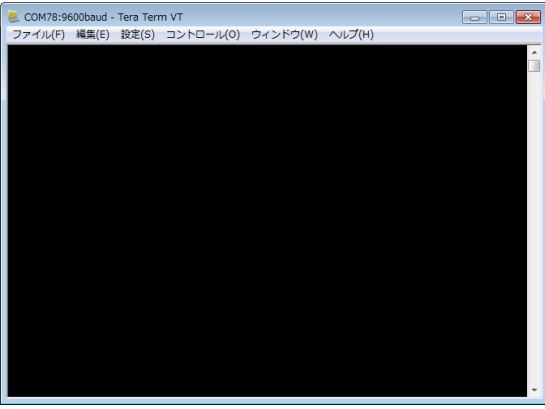
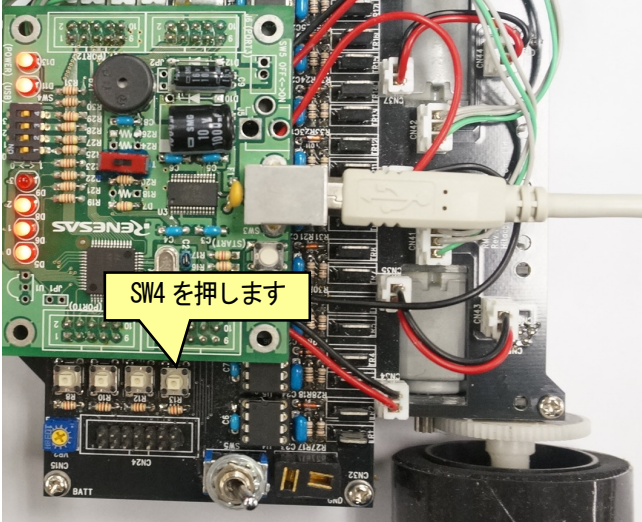
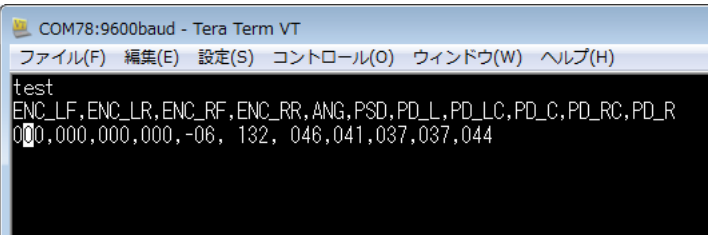
2		<p>①基板マイコンカーの電源をONにします。</p> <p>②ルネサス統合開発環境「ツール→R8C Writer」で書き込みソフトを起動して、動作確認プログラムを書き込みます。</p>
---	--	---

3	 <p>②書き込みスイッチは左側(実行側)</p> <p>①電源スイッチは左側(OFF側)</p>	<p>正常に書き込みが終わったら、</p> <ol style="list-style-type: none"> ①基板マイコンカーの電源をOFFにします。 ②マイコンボードの書き込みスイッチは左側(プログラム実行側)にします。
---	--	--

3.4 Tera Term を使った動作確認

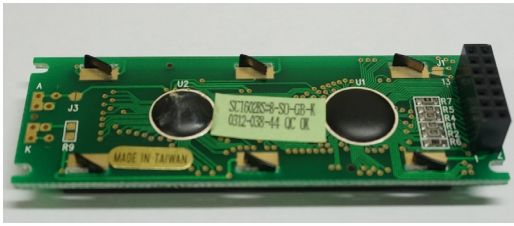
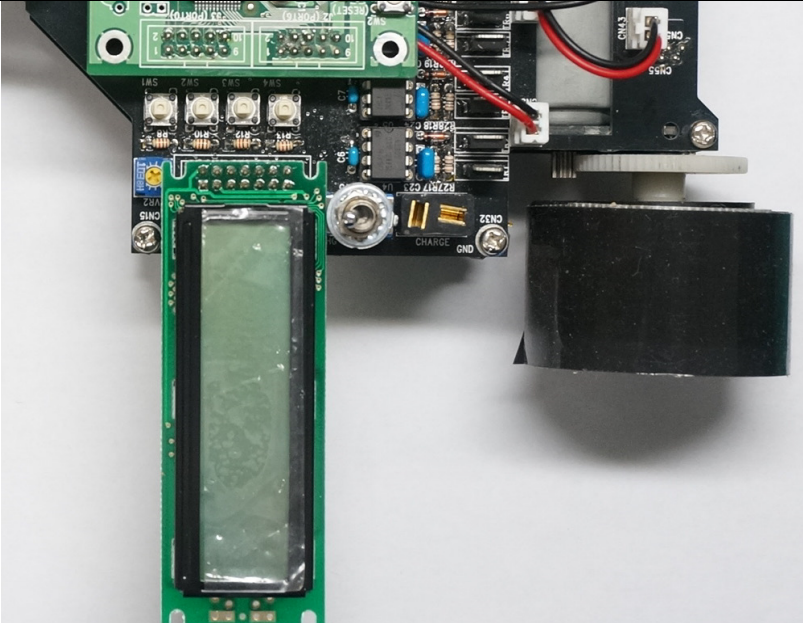
1	 <p>書き込みスイッチは左側(実行側)のまま</p> <p>①USBケーブルはパソコンと接続</p> <p>②電源スイッチは右側(ON側)</p>	<ol style="list-style-type: none"> ①USBケーブルをパソコンに接続します。 ②電源スイッチを右側(ON側)にします。
---	--	--

2	 <p>①シリアルポートを選択</p> <p>②OK</p>	<p>Tera Term を立ち上げます。</p> <ol style="list-style-type: none"> ①シリアルポートを選択します。ポート番号は、基板マイコンカーのマイコンボードの番号を選択します。 ②「OK」をクリックします。
---	---	--

<p>3</p>		<p>Tera Term がシリアルポートと接続されました。</p>
<p>4</p>		<p>基板マイコンカーの SW4 を押すと、Tera Term の画面内に、基板マイコンカーの状態が表示され続けます。</p>
<p>5</p>	 <pre> COM78:9600baud - Tera Term VT ファイル(F) 編集(E) 設定(S) コントロール(O) ウィンドウ(W) ヘルプ(H) test ENC_LF,ENC_LR,ENC_RF,ENC_RR,ANG,PSD,PD_L,PD_LC,PD_C,PD_RC,PD_R 000,000,000,000,-06, 132, 046,041,037,037,044 </pre>	<p>Tera Term に、基板マイコンカーの状態が表示され続けます。</p>

4. 液晶の使い方

4.1 液晶の接続

1		<p>左写真のように、液晶付属の14ピンメスコネクタを液晶に半田付けします。</p>
2		<p>液晶は写真のように接続します。 走行するときは、液晶は取り外します。</p>

4.2 液晶を制御する関数

4.2.1 液晶の初期化

書式	<code>int init_lcd(void);</code>
内容	液晶を初期化します。
引数	なし
戻り値	0:異常 1:正常
使用例	<pre>asm(" fset I "); /* 全体の割り込み許可 */ init_lcd(); /* LCD 初期化 */</pre> <p>※init_lcd 関数は、全体割り込み許可をした後に実行してください。</p>

4.2.2 液晶表示処理

書式	<code>void lcd_process(void);</code>
内容	液晶の制御を行います。この関数は割り込み処理などで 1ms ごとに実行してください。
引数	なし
戻り値	なし
使用例	<pre>/* LCD 表示処理用関数(1ms ごとに実行) */ lcd_process();</pre> <p>液晶の表示は、1文字表示に最大で 10ms の時間がかかります(データシート参照)。32文字表示しようとする、最大で 320ms かかってしまいます。1回の表示に 320ms も時間を取られては他の処理が何もできなくなり大問題です。</p> <p>lcd_process 関数は 1ms ごとに実行して、1ms ごとに少しずつ液晶表示処理をする関数です。この関数は 1ms ごとに実行してください。今回は、タイマ RB の 1ms ごとの割り込み関数内で実行します。</p> <p>【この関数の仕組み】 本来は、1文字分の表示データを液晶に送った後、10ms 待つてから次の表示データを送らなければいけません。 この関数は 1文字分の表示データを液晶に送った後、すぐに関数を終了します。次にこの関数が実行されても、まだ 1ms しか経っていないので何もせずに関数を終了させます。10回目で、次の 1文字分のデータを送り、またすぐに関数を終了します。これを繰り返します。</p>

4.2.3 液晶表示処理

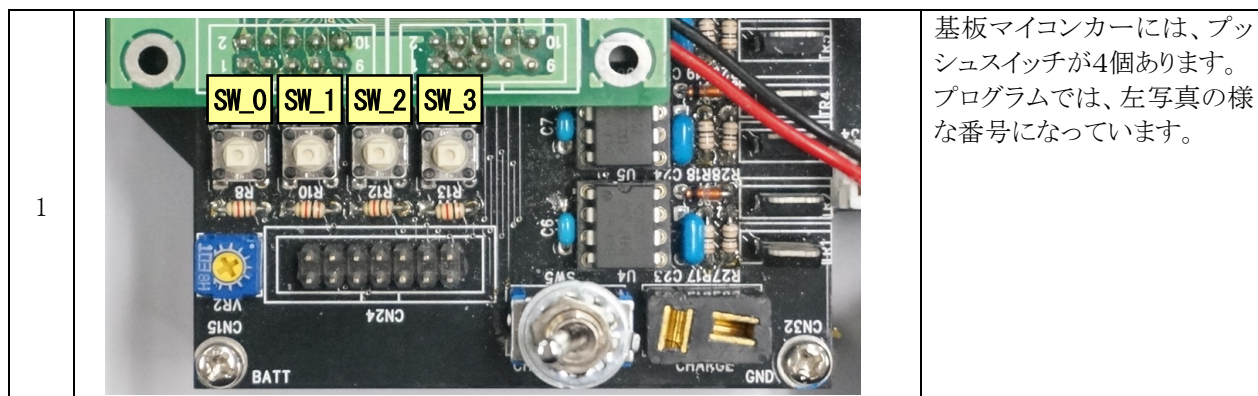
書式	<code>int lcd_printf(char far *format, ...);</code>																																																						
内容	液晶に文字を表示します。書式は、 <code>printf</code> 関数と同じです。表示位置は前回表示された続きか、 <code>lcd_position</code> 関数で指定された位置です。																																																						
引数	<code>char *format</code> 書式文字列 ... 可変個引数																																																						
戻り値	正常時:出力した文字列 異常時:負の数																																																						
使用例	<pre>lcd_printf("0123456789abcdef"); lcd_printf("0123456789ABCDEF");</pre> <p>実行すると下図のようになります。</p> <table border="1"> <tr> <td></td> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td> <td>列</td> </tr> <tr> <td>0 行</td> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>a</td><td>b</td><td>c</td><td>d</td><td>e</td><td>f</td> <td></td> </tr> <tr> <td>1 行</td> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td><td>F</td> <td></td> </tr> </table>		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	列	0 行	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f		1 行	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	列																																						
0 行	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f																																							
1 行	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F																																							

4.2.4 液晶表示処理

書式	<code>void lcd_position(char x ,char y);</code>																																																						
内容	液晶に表示する位置を指定します。																																																						
引数	<code>char x</code> 列 (0 ~ 15) <code>char y</code> 行 (0 ~ 1)																																																						
戻り値	なし																																																						
使用例	<pre>lcd_position(0, 0); /* 表示する位置を指定 */ lcd_printf("LCD/microSD PCB "); /* 表示する文字列 */</pre> <p>実行すると下図のようになります。</p> <table border="1"> <tr> <td></td> <td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td> <td>列</td> </tr> <tr> <td>0 行</td> <td>L</td><td>C</td><td>D</td><td>/</td><td>m</td><td>i</td><td>c</td><td>r</td><td>o</td><td>S</td><td>D</td><td></td><td>P</td><td>C</td><td>B</td><td></td> <td></td> </tr> <tr> <td>1 行</td> <td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td> <td></td> </tr> </table>		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	列	0 行	L	C	D	/	m	i	c	r	o	S	D		P	C	B			1 行																	
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	列																																						
0 行	L	C	D	/	m	i	c	r	o	S	D		P	C	B																																								
1 行																																																							

5. プッシュスイッチの使い方

5.1 プッシュスイッチ



5.2 プッシュスイッチを制御する関数

5.2.1 プッシュスイッチ関係処理の初期化

書式	<code>void init_switch(void);</code>
内容	プッシュスイッチを初期化します。
引数	なし
戻り値	なし
使用例	<pre>asm(" fset I "); /* 全体の割り込み許可 */ init_switch(); /* スイッチ初期化 */</pre> <p>※init_switch 関数は、全体割り込み許可をした後に実行してください。</p>

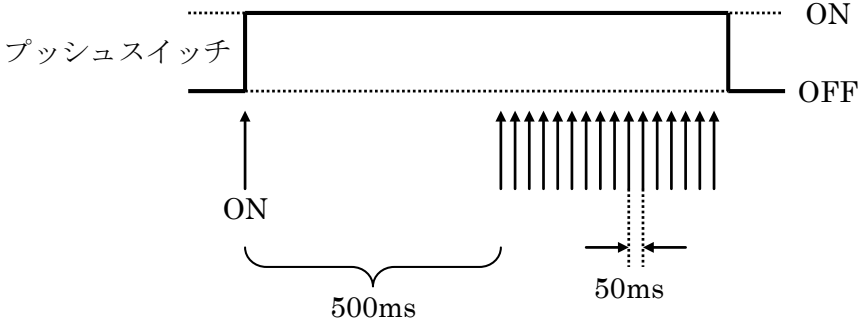
5.2.2 プッシュスイッチ検出処理

書式	<code>void switch_process(void);</code>
内容	プッシュスイッチの制御を行います。この関数は割り込み処理などで 1ms ごとに実行してください。
引数	なし
戻り値	なし
使用例	<pre>/* プッシュスイッチ検出処理関数(1ms ごとに実行) */ switch_process();</pre> <p>実際に、プッシュスイッチ処理をしているのは、swich_process 関数です。この関数は、1ms ごとに実行してください。今回は、タイマ RB の 1ms ごとの割り込み関数内で実行します。</p>

5.2.3 プッシュスイッチの現在情報取得

書式	<code>unsigned char get_sw_now(void);</code>
内容	プッシュスイッチの現在値を取得します。
引数	なし
戻り値	0:スイッチ OFF 1:スイッチ ON
使用例	<pre> if(get_sw_now() & SW_0) { // SW_0部分に、SW_0~SW_3を入れます SW_0が"1"(OFF)なら } else { SW_0が"0"(ON)なら } </pre>

5.2.4 プッシュスイッチのフラグ情報取得

書式	<code>unsigned char get_sw_flag(unsigned char flag);</code>
内容	プッシュスイッチのキーリピート後の値を取得します。
引数	取得するキー SW_0 ~ SW_3
戻り値	<p>キーリピート処理後の値 0:OFF 0以外:ON</p> <p>時計などの時刻を設定するとき、押した瞬間+1だけして、そのまま押し続けると連続して値がプラスされていく仕組みがあります。この関数はその機能を実現します。</p> <p>get_sw_flag 関数では、次のような動作になります。</p>  <p>プッシュスイッチが押された瞬間、ONになります。その後0.5秒押し続けるとリピート機能が働いて、50msごとにON信号が送られてきます。</p>
使用例	<pre> if(get_sw_flag(SW_0)) { // SW_0部分に、SW_0~SW_3を入れます SW_1がONなら } else { SW_1がOFFなら } </pre>