赤外線受光モジュール
C 言語リモコンプログラム
解説マニュアル

第 1.00 版
2010 年 4 月 26 日
ルネサスマイコンカーラリー事務局
日立インターメディックス株式会社

注 意 事 項 (rev.2.0)

著作権

・本マニュアルに関する著作権はルネサスマイコンカーラリー事務局に帰属します。
・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、文書によるルネサスマイコンカーラリー事務局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、ルネサスマイコンカーラリー事務局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、ルネサスマイコンカーラリー事務局は、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

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

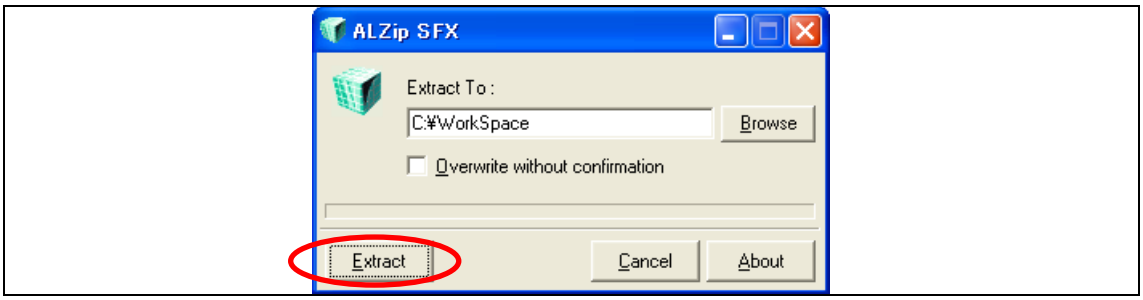
1. 概要

本書では、赤外線受光モジュールを使用した、C 言語リモコンプログラムの解説を行います。
リモコンプログラムは、ミニマイコンカーVer.2 の赤外線受光モジュールを使用して、市販の赤外線リモコンでミニマイコンカーの操作をおこなうことができるようにするプログラムです。
開発環境の構築方法やプログラムのビルド、書き込みについては、「ミニマイコンカー製作キット Ver.2　C 言語走行プログラム解説マニュアル」の「3. インストール」「4. ミニマイコンカーVer.2 の動作確認」を参照してください。

2. ワークスペースのインストール

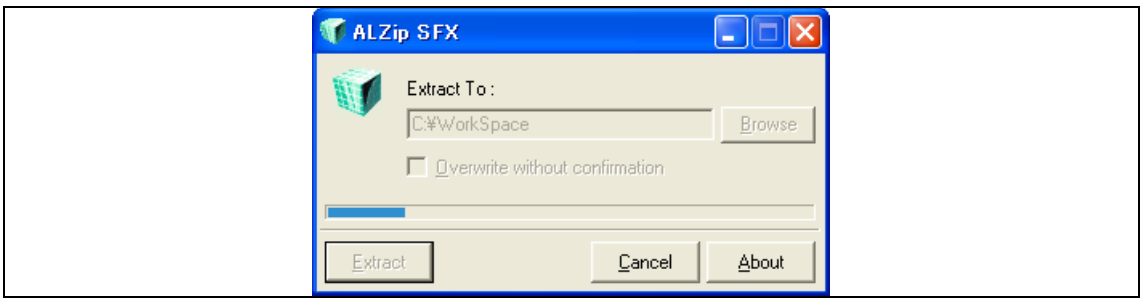
日立インターメディックス株式会社のマイコンカーラリー販売ページからワークスペースのインストーラー「mini_mcr2_ir_vxxx.exe」（xxx はバージョン）をダウンロードします。

ダウンロードした「mini_mcr2_ir_vxxx.exe」をダブルクリックし、インストーラーを実行します。

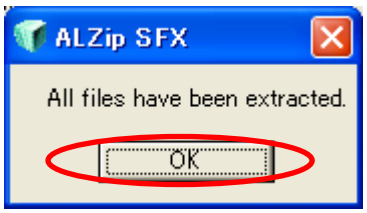


表示されたデフォルトのインストール先のフォルダ「c:¥WorkSpace」を確認して、「Extract」をクリックします。

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



インストールが開始されます。



ワークスペースのインストールが完了しました。「OK」をクリックします。

以上でワークスペースのインストールは完了です。

3. プログラム解説「mini_mcr.c」

WorkSpace のファイルをそのままコンパイルして書き込むと、赤外線リモコン（NEC フォーマットの日立地上アナログテレビ）の「2」「4」「6」「8」ボタンでミニマイコンカーVer.2 を操作できます。

3.1　プログラムリスト

```
 1 :  //---------------------------------------------------------------------------
 2 :  // 対象マイコン R8C/35A
 3 :  // ﾌｧｲﾙ内容     リモコンプログラム
 4 :  // バージョン   Ver.1.00
 5 :  // Date         2010.4.20
 6 :  // Copyright    ルネサスマイコンカーラリー事務局
 7 :  //              日立インターメディックス株式会社
 8 :  //---------------------------------------------------------------------------
 9 :  //---------------------------------------------------------------------------
10 :  // インクルード
11 :  //---------------------------------------------------------------------------
12 :  #include <stdio.h>
13 :  #include <string.h>
14 :  #include "sfr_r835a.h"
15 :  #include "printf_lib.h"
16 :
17 :  //---------------------------------------------------------------------------
18 :  // シンボル定義
19 :  //---------------------------------------------------------------------------
20 :  #define TIMER_CYCLE     155             // 1ms:0.001/(1/(20000000/128))-1
21 :  #define PWM_CYCLE       39999           // 16ms:0.016/(1/(20000000/8))-1
22 :
23 :  #define Def_500Hz       4999            // 500Hz:(1/500)/(1/(20000000/8))-1
24 :  #define Def_1000Hz      2499            // 1000Hz:(1/1000)/(1/(20000000/8))-1
25 :
26 :  #define Def_C3          19083           // ド:(1/131)/(1/(20000000/8))-1
27 :  #define Def_D3          17006           // レ:(1/147)/(1/(20000000/8))-1
28 :  #define Def_E3          15151           // ミ:(1/165)/(1/(20000000/8))-1
29 :  #define Def_F3          14285           // ファ:(1/175)/(1/(20000000/8))-1
30 :  #define Def_G3          12754           // ソ:(1/196)/(1/(20000000/8))-1
31 :  #define Def_A3          11362           // ラ:(1/220)/(1/(20000000/8))-1
32 :  #define Def_B3          10120           // シ:(1/247)/(1/(20000000/8))-1
33 :  #define Def_C4          9541            // ド:(1/262)/(1/(20000000/8))-1
34 :
```

```
 35 :  #define DI()            asm("FCLR I")   // 割り込み禁止
 36 :  #define EI()            asm("FSET I")   // 割り込み許可
 37 :
 38 :  #define DefIrS          11249           // リーダーコード(4.5ms):0.0045/(1/(20000000/8))-1
 39 :  #define DefIrR          5624            // リピートリーダーコード(2.25ms):0.00225/(1/(20000000/8))-1
 40 :  #define DefIr0          1412            // 0(0.565ms):0.000565/(1/(20000000/8))-1
 41 :  #define DefIr1          4224            // 1(1.69ms):0.00169/(1/(20000000/8))-1
 42 :  #define DefIrM          249             // マージン(0.1ms):0.0001/(1/(20000000/8))-1
 43 :  //-----------------------------------------------------------------------------
 44 :  // 関数プロトタイプの宣言
 45 :  //-----------------------------------------------------------------------------
 46 :  void init( void );
 47 :  unsigned char sensor( void );
 48 :  void motor( int data1, int data2 );
 49 :  void timer( unsigned long timer_set );
 50 :  void beep( int data1 );
 51 :  unsigned char dipsw( void );
 52 :  unsigned char pushsw( void );
 53 :
 54 :  //-----------------------------------------------------------------------------
 55 :  // グローバル変数の宣言
 56 :  //-----------------------------------------------------------------------------
 57 :  unsigned long   cnt0 = 0;               // timer 関数用
 58 :  unsigned long   cnt1 = 0;               // main 内で使用
 59 :
 60 :  volatile char   ir_data[64];
 61 :  //-----------------------------------------------------------------------------
 62 :  // メインプログラム
 63 :  //-----------------------------------------------------------------------------
 64 :  void main(void)
 65 :  {
 66 :          int i;
 67 :          int button;
 68 :
 69 :          // 初期化
 70 :          init();
 71 :          init_uart0_printf(SPEED_9600);
 72 :
 73 :          while(1){
 74 :
 75 :                  // 赤外線の受信データのリーダーコード、リピートリーダーコードをクリア
 76 :                  ir_data[0] = ' ';
 77 :
 78 :                  // 受信のため待ち
 79 :                  timer(200);
 80 :
 81 :                  // リーダーコード、リピートリーダーコードが受信されていなかった場合
 82 :                  if( ir_data[0] == ' ' ){
 83 :                          // データをすべてクリア
 84 :                          for(i=0;i<60;i++){
 85 :                                  ir_data[i] = ' ';
 86 :                          }
 87 :                  }
 88 :
 89 :                  // ボタンが押されていない
 90 :                  button = 0;
 91 :  #if 1
 92 :  //139 日立 地上アナログテレビ
 93 :                  // 2 ボタン
 94 :                  if( strncmp( &ir_data[0], "S0000101011110101011100001000111", 1+32 ) == 0 ||
 95 :                          strncmp( &ir_data[0], "R0000101011110101011100001000111", 1+32 ) == 0  ){
 96 :                          button = 2;
 97 :                  }
 98 :                  // 4 ボタン
 99 :                  if( strncmp( &ir_data[0], "S0000101011110101001110001100111", 1+32 ) == 0 ||
100 :                          strncmp( &ir_data[0], "R0000101011110101001110001100111", 1+32 ) == 0  ){
101 :                          button = 4;
102 :                  }
103 :                  // 6 ボタン
104 :                  if( strncmp( &ir_data[0], "S0000101011110101011100010000111", 1+32 ) == 0 ||
105 :                          strncmp( &ir_data[0], "R0000101011110101011100010000111", 1+32 ) == 0  ){
106 :                          button = 6;
```

```
107 :                   }
108 :                   // 8ボタン
109 :                   if( strncmp( &ir_data[0], "S0000101011110101001000011011111", 1+32 ) == 0 ||
110 :                           strncmp( &ir_data[0], "R0000101011110101001000011011111", 1+32 ) == 0  ){
111 :                           button = 8;
112 :                   }
113 :   #else
114 :   //134 NEC 地上アナログテレビ
115 :                   // 2ボタン
116 :                   if( strncmp( &ir_data[0], "S0001100011100111100010000110111", 1+32 ) == 0 ||
117 :                           strncmp( &ir_data[0], "R0001100011100111100010000110111", 1+32 ) == 0 ){
118 :                           button = 2;
119 :                   }
120 :                   // 4ボタン
121 :                   if( strncmp( &ir_data[0], "S0001100011100111110010000110111", 1+32 ) == 0 ||
122 :                           strncmp( &ir_data[0], "R0001100011100111110010000110111", 1+32 ) == 0 ){
123 :                           button = 4;
124 :                   }
125 :                   // 6ボタン
126 :                   if( strncmp( &ir_data[0], "S0001100011100111101010000101011", 1+32 ) == 0 ||
127 :                           strncmp( &ir_data[0], "R0001100011100111101010000101011", 1+32 ) == 0 ){
128 :                           button = 6;
129 :                   }
130 :                   // 8ボタン
131 :                   if( strncmp( &ir_data[0], "S0001100011100111111010000010111", 1+32 ) == 0 ||
132 :                           strncmp( &ir_data[0], "R0001100011100111111010000010111", 1+32 ) == 0 ){
133 :                           button = 8;
134 :                   }
135 :   #endif
136 :
137 :                   switch( button ){
138 :                   // ボタンが押されていないか
139 :                   case 0:
140 :                           motor( 0, 0 );
141 :                           break;
142 :
143 :                   // 2ボタンで前進
144 :                   case 2:
145 :                           motor( 100, 100 );
146 :                           break;
147 :
148 :                   // 4ボタンで左旋回
149 :                   case 4:
150 :                           motor( 0, 100 );
151 :                           break;
152 :
153 :                   // 6ボタンで右旋回
154 :                   case 6:
155 :                           motor( 100, 0 );
156 :                           break;
157 :
158 :                   // 8ボタンでバック
159 :                   case 8:
160 :                           motor( -100, -100 );
161 :                           break;
162 :
163 :                   default:
164 :                           break;
165 :
166 :                   }
167 :
168 :                   // デバッグ出力
169 :                   for(i=0;i<60;i++){
170 :                           printf("%c",ir_data[i]);
171 :                   }
172 :                   printf("¥n");
173 :                   printf("¥n");
174 :
175 :           }
176 :
177 :   }
178 :
```

```
179 :  //------------------------------------------------------------------------
180 :  // R8C/35A の内蔵周辺機能の初期化
181 :  //------------------------------------------------------------------------
182 :  void init( void )
183 :  {
184 :          unsigned char i = 0;
185 :
186 :          // 割り込み禁止
187 :          DI();
188 :
189 :          // クロック発生回路の XIN クロック設定
190 :          prc0 = 1;
191 :
192 :          cm13 = 1;
193 :          cm05 = 0;
194 :          while(i <= 50) i++;
195 :          ocd2 = 0;
196 :
197 :          prc0 = 0;
198 :
199 :          // I/O ポートの入出力設定
200 :          prc2 = 1;                       // pd0 レジスタへの書き込み許可
201 :          pd0 = 0xe0;                     // P0_0～P0_3:センサー
202 :                                          // P0_4:マイクロスイッチ
203 :                                          // P0_5～P0_7:LED
204 :          prc2 = 0;                       // pd0 レジスタへの書き込み禁止
205 :          pd1 = 0xdf;                     // P1_0～P1_3:LED
206 :                                          // P1_4:TXD0
207 :                                          // P1_5:RXD0
208 :          pd2 = 0xfe;                     // P2_0:スイッチ
209 :                                          // P2_1:AIN1
210 :                                          // P2_2:PWMA
211 :                                          // P2_3:BIN1
212 :                                          // P2_4:PWMB
213 :                                          // P2_5:SERVO
214 :                                          // P2_6:AIN2
215 :                                          // P2_7:BIN2
216 :          pd3 = 0xfb;                     // P3_2:赤外線受信
217 :                                          // P3_4:ブザー
218 :          pd4 = 0x80;                     // P4_2:VREF
219 :                                          // P4_3～P4_5:DIPSW
220 :                                          // P4_6:XIN
221 :                                          // P4_7:XOUT
222 :          pd5 = 0x40;                     // P5_7:DIPSW
223 :          pd6 = 0xff;                     // P6_7:¥ENABLE¥
224 :                                          // P6_6:LATCH
225 :                                          // P6_5:¥SCLR¥
226 :                                          // P6_4:SCLK
227 :                                          // P6_3:ROWA_SIN        行 A
228 :                                          // P6_2:ROWB_SIN        行 B
229 :                                          // P6_1:COL_SIN         列
230 :                                          // P6_0:
231 :
232 :
233 :
234 :          mstcr = 0x00;                   // モジュールストップ解除
235 :
236 :          // タイマーRA のパルス幅測定モードで赤外線リモコンの受信パルスを測定
237 :  //      traioc = 0x00;                  // L レベル幅、フィルタなし
238 :  //      traioc = 0x10;                  // L レベル幅、フィルタあり
239 :  //      traioc = 0x01;                  // H レベル幅、フィルタなし
240 :          traioc = 0x11;                  // H レベル幅、フィルタあり
241 :
242 :          tramr = 0x13;                   // パルス幅測定モード f8
243 :          trapre = 0xff;                  //
244 :          tra = 0xff;                     //
245 :          trasr = 0x03;                   // P3_2 に割り当てる
246 :          traic = 0x01;                   // タイマーRA の割り込みレベル設定
247 :          tracr = 0x01;                   // スタート
248 :
249 :          // タイマーRB の 1ms 割り込み設定
250 :          trbmr = 0x00;                   // カウントソースは f1
```

```
251 :          trbpre = 128 - 1;               // プリスケーラ
252 :          trbpr = TIMER_CYCLE;            // プライマリカウンタ
253 :          trbic = 0x01;                   // タイマーRB の割り込みレベル設定
254 :          trbcr = 0x01;                   // カウントを開始
255 :
256 :          // タイマーRC の PWM モード
257 :          trccr1 = 0xb0;                  // カウントソースは f8
258 :          trcgra = 0;                     // 圧電サウンダの周期
259 :          trcgrc = 0;                     // 圧電サウンダのデューティ比
260 :          trccr2 = 0x02;                  // TRCIOC 端子はアクティブレベル H
261 :          trcoer = 0x0b;                  // TRCIOC 端子の出力許可
262 :          trcpsr1 = 0x02;                 // TRCIOC 端子を P3_4 に割り当て
263 :          trcmr = 0x8a;                   // カウントを開始
264 :
265 :          // タイマーRD のリセット同期 PWM モード
266 :          trdpsr0 = 0x08;                 // TRDIOB0 端子を P2_2 に割り当て
267 :          trdpsr1 = 0x05;                 // TRDIOB1 端子を P2_5 に割り当て
268 :                                          // TRDIOA1 端子を P2_4 に割り当て
269 :          trdmr = 0xf0;                   // レジスタをバッファ動作にする
270 :          trdfcr = 0x01;                  // リセット同期 PWM モードに設定
271 :          trdoer1 = 0xcd;                 // TRDIOB1 の出力許可
272 :                                          // TRDIOA1 の出力許可
273 :                                          // TRDIOB0 端子の出力許可
274 :          trdcr0 = 0x23;                  // カウントソースは f8
275 :          trdgra0 = trdgrc0 = PWM_CYCLE;  // 周期
276 :          trdgrb0 = trdgrd0 = 0;          // TRDIOB0 端子（左モーター）
277 :          trdgra1 = trdgrc1 = 0;          // TRDIOA1 端子（右モーター）
278 :          trdgrb1 = trdgrd1 = 0;          // TRDIOB1 端子（サーボ）
279 :          trdstr = 0x0d;                  // カウントを開始
280 :
281 :          // 割り込み許可
282 :          EI();
283 :  }
284 :
285 :  //-----------------------------------------------------------------------
286 :  // 割り込み
287 :  //-----------------------------------------------------------------------
288 :
289 :  #pragma interrupt intTRAIC (vect=22)
290 :  void intTRAIC( void )
291 :  {
292 :          static int      num = -1;
293 :          unsigned short  data;
294 :          int             i;
295 :
296 :          // アンダーフロー
297 :          if( tundf_tracr == 1){
298 :
299 :                  // アンダーフローフラグをクリア
300 :                  tundf_tracr = 0;
301 :
302 :                  tstart_tracr = 0;       // カウントストップ
303 :                  tra = 0xff;             // カウンタをリセット
304 :                  trapre = 0xff;          // カウンタをリセット
305 :                  tstart_tracr = 1;       // カウントスタート
306 :
307 :          }
308 :
309 :          // エッジ検出
310 :          if( tedgf_tracr == 1){
311 :
312 :                  // エッジ検出フラグをクリア
313 :                  tedgf_tracr = 0;
314 :
315 :                  // カウンタ値を取得
316 :                  data = tra;
317 :                  data = data << 8;
318 :                  data = data + trapre;
319 :                  data = 65535 - data;
320 :
321 :                  // リーダーコード検出
322 :                  if( ( DefIrS - DefIrM ) < data && data < ( DefIrS + DefIrM ) ){
```

```
323 :                           num = 0;
324 :                           ir_data[0] = 'S';
325 :                   }
326 :
327 :                   // 0 受信
328 :                   if( ( DefIr0 - DefIrM ) < data && data < ( DefIr0 + DefIrM ) && 0 <= num && num <= 31 ){
329 :                           num++;
330 :                           ir_data[num] = '0';
331 :                   }
332 :
333 :                   // 1 受信
334 :                   if( ( DefIr1 - DefIrM ) < data && data < ( DefIr1 + DefIrM ) && 0 <= num && num <= 31 ){
335 :                           num++;
336 :                           ir_data[num] = '1';
337 :                   }
338 :
339 :                   // リピートリーダーコード検出
340 :                   if( ( DefIrR - DefIrM ) < data && data < ( DefIrR + DefIrM ) && num == 32 ){
341 :                           ir_data[0] = 'R';
342 :                   }
343 :
344 :                   tstart_tracr = 0;       // カウントストップ
345 :                   tra = 0xff;             // カウンタをリセット
346 :                   trapre = 0xff;          // カウンタをリセット
347 :                   tstart_tracr = 1;       // カウントスタート
348 :
349 :           }
350 :
351 :   }
352 :
353 :   #pragma interrupt intTRBIC (vect=24)
354 :   void intTRBIC( void )
355 :   {
356 :           tundf_tracr = 0;
357 :
358 :           p0_7 = ~p0_7;
359 :
360 :           if( p0_7 == 0 ){
361 :                   //p0_1、p0_3 のモニタが可能
362 :                   p0_5 = ~p0_1;
363 :                   p0_6 = ~p0_3;
364 :           }else{
365 :                   //p0_0、p0_2 のモニタが可能
366 :                   p0_5 = p0_0;
367 :                   p0_6 = p0_2;
368 :           }
369 :
370 :           cnt0++;
371 :           cnt1++;
372 :
373 :   }
374 :
375 :   //--------------------------------------------------------------------------------
376 :   // センサー状態検出
377 :   // 引数         なし
378 :   // 戻り値       センサ値
379 :   //--------------------------------------------------------------------------------
380 :   unsigned char sensor( void )
381 :   {
382 :           volatile unsigned char  data1;
383 :
384 :           data1 = ~p0;                    // ラインの色は白
385 :           data1 = data1 & 0x0f;
386 :
387 :           return( data1 );
388 :   }
389 :
390 :   //--------------------------------------------------------------------------------
391 :   // モーター速度制御
392 :   // 引数         左モーター:-100～100、右モーター:-100～100
393 :   //              0 で停止、100 で正転 100%、-100 で逆転 100%
394 :   // 戻り値       なし
```

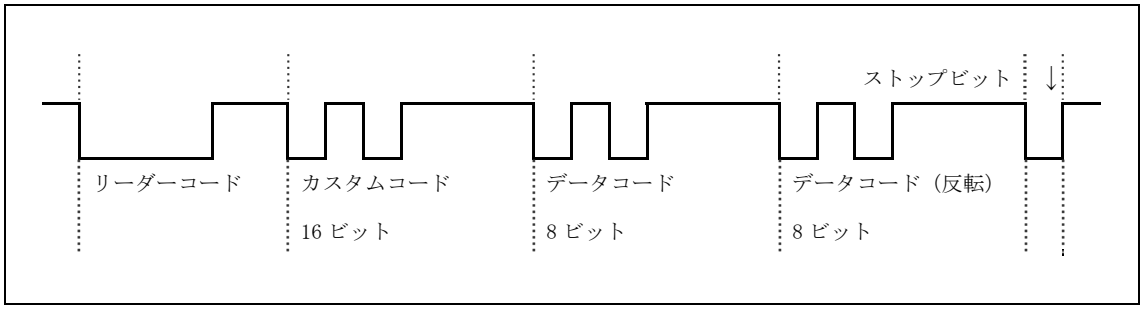
```
395 :  //---------------------------------------------------------------------------
396 :  void motor( int data1, int data2 )
397 :  {
398 :          volatile int    motor_r;
399 :          volatile int    motor_l;
400 :          volatile int    sw_data;
401 :
402 :          sw_data = dipsw() + 5;
403 :          motor_l = (long)data1 * sw_data / 20;
404 :          motor_r = (long)data2 * sw_data / 20;
405 :
406 :          if( motor_l >= 0 ) {
407 :                  p2_1 = 0;
408 :                  p2_6 = 1;
409 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
410 :          } else {
411 :                  p2_1 = 1;
412 :                  p2_6 = 0;
413 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
414 :          }
415 :
416 :          if( motor_r >= 0 ) {
417 :                  p2_3 = 0;
418 :                  p2_7 = 1;
419 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
420 :          } else {
421 :                  p2_3 = 1;
422 :                  p2_7 = 0;
423 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
424 :          }
425 :  }
426 :
427 :  //---------------------------------------------------------------------------
428 :  // 時間稼ぎ
429 :  // 引数          タイマー値 1=1ms
430 :  // 戻り値        なし
431 :  //---------------------------------------------------------------------------
432 :  void timer( unsigned long data1 )
433 :  {
434 :          cnt0 = 0;
435 :          while( cnt0 < data1 );
436 :  }
437 :
438 :  //---------------------------------------------------------------------------
439 :  // 音を鳴らす
440 :  // 引数          (1/音の周波数)/(1/(クロック周波数/8))-1
441 :  // 戻り値        なし
442 :  //---------------------------------------------------------------------------
443 :  void beep( int data1 )
444 :  {
445 :          trcgra = data1;                 // 周期の設定
446 :          trcgrc = data1 / 2;             // デューティ 50%のため周期の半分の値
447 :  }
448 :
449 :  //---------------------------------------------------------------------------
450 :  // DIP スイッチ状態検出
451 :  // 引数          なし
452 :  // 戻り値        0～15、DIP スイッチが ON の場合、対応するビットが 0 になります。
453 :  //---------------------------------------------------------------------------
454 :  unsigned char dipsw( void )
455 :  {
456 :          volatile unsigned char  data1;
457 :
458 :          data1 = ( ( p5 >> 4 ) & 0x08 ) | ( ( p4 >> 3 ) & 0x07 );
459 :
460 :          return( data1 );
461 :  }
462 :
463 :  //---------------------------------------------------------------------------
464 :  // プッシュスイッチ状態検出
465 :  // 引数          なし
466 :  // 戻り値        スイッチが押されていない場合:0、押された場合:1
```

```
467 :  //--------------------------------------------------------------------------
468 :  unsigned char pushsw( void )
469 :  {
470 :          unsigned char data1;
471 :
472 :          data1 = ~p2;
473 :          data1 &= 0x01;
474 :
475 :          return( data1 );
476 :  }
```

3.2　赤外線リモコンのフォーマットについて

本書およびプログラムでは、NEC フォーマットの赤外線リモコンを使用しています。

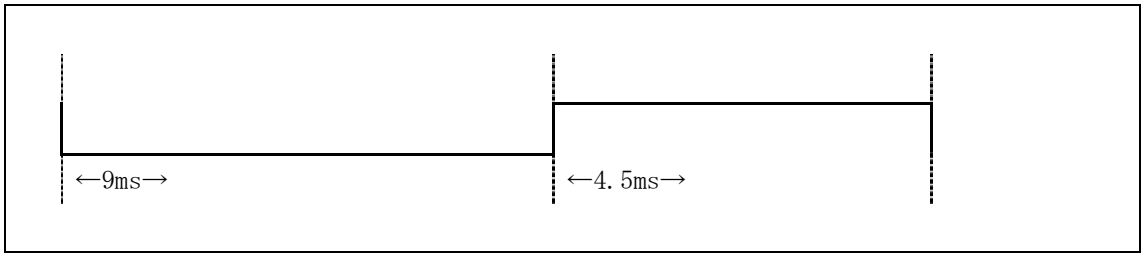
3.2.1　フレーム構成



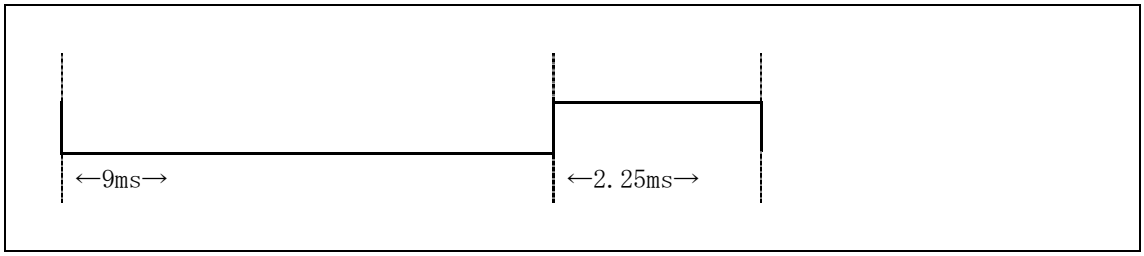
リーダーコードと 32 ビットのデータ、ストップビットで 1 フレームが構成されています。
ボタンを押しっぱなしにした場合、2 フレーム目からは、リーダーコード（リピート）とストップビットのみになります。これらのコードの H レベル幅を測定し、コードの種類の判断をおこないます。

3.2.2　リーダーコード

1 回目

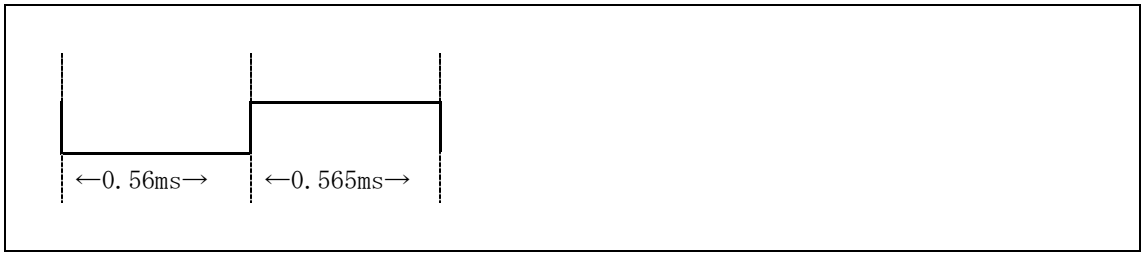


リピート

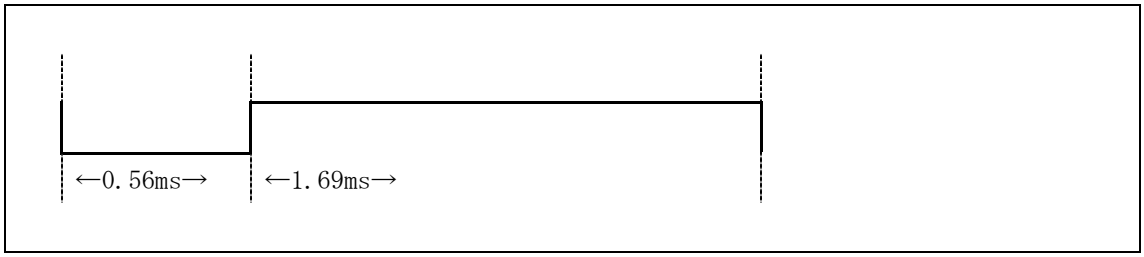


3.2.3　カスタムコード、データコード、データコード（反転）

0



1



3.3　シンボル定義

プログラム

```
38 :  #define DefIrS          11249       // リーダーコード(4.5ms):0.0045/(1/(20000000/8))-1
39 :  #define DefIrR          5624        // リピートリーダーコード(2.25ms):0.00225/(1/(20000000/8))-1
40 :  #define DefIr0          1412        // 0(0.565ms):0.000565/(1/(20000000/8))-1
41 :  #define DefIr1          4224        // 1(1.69ms):0.00169/(1/(20000000/8))-1
42 :  #define DefIrM          249         // マージン(0.1ms):0.0001/(1/(20000000/8))-1
```

名称	説明
DefIrS	DefIrS はリーダーコード（1 回目）の H レベル幅を設定します。 今回は 4.5［ms］に設定しますので、 $4.5\times10^{-3}\div(1\div(20\times10^{6}\div8))-1=11249$
DefIrR	DefIrR はリーダーコード（リピート）の H レベル幅を設定します。 今回は 2.25［ms］に設定しますので、 $2.25\times10^{-3}\div(1\div(20\times10^{6}\div8))-1=5624$
DefIr0	DefIr0 は 0 の H レベル幅を設定します。 今回は 0.565［ms］に設定しますので、 $0.565\times10^{-3}\div(1\div(20\times10^{6}\div8))-1=1412$
DefIr1	DefIr1 は 1 の H レベル幅を設定します。 今回は 1.69［ms］に設定しますので、 $1.69\times10^{-3}\div(1\div(20\times10^{6}\div8))-1=4224$
DefIrM	DefIrM は H レベル幅のマージンを設定します。 今回は 0.1［ms］に設定しますので、 $0.1\times10^{-3}\div(1\div(20\times10^{6}\div8))-1=249$

3.4　メインプログラムを説明する前に

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

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

3.5　R8C/35A の内蔵周辺機能の初期化：init 関数

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

```
236 :          // タイマーRA のパルス幅測定モードで赤外線リモコンの受信パルスを測定
237 :  //      traioc = 0x00;                  // L レベル幅、フィルタなし
238 :  //      traioc = 0x10;                  // L レベル幅、フィルタあり
239 :  //      traioc = 0x01;                  // H レベル幅、フィルタなし
240 :          traioc = 0x11;                  // H レベル幅、フィルタあり
241 :
242 :          tramr = 0x13;                   // パルス幅測定モード f8
243 :          trapre = 0xff;                  //
244 :          tra = 0xff;                     //
245 :          trasr = 0x03;                   // P3_2 に割り当てる
246 :          traic = 0x01;                   // タイマーRA の割り込みレベル設定
247 :          tracr = 0x01;                   // スタート
```

タイマーRA をパルス幅測定モードにして使用するために設定をおこないます。

レジスタ	ビット	シンボル	説明	設定値
TRAIOC	7	TIOGT1	パルス幅測定モードでは“0”にします。	0x11
	6	TIOGT0		
	5	TIPF1	TRAIO 端子の入力フィルタを f1 にするため“1”にします。	
	4	TIPF0		
	3	TIOSEL	ハードウェア LIN 機能は使用しないので“0”にします。	
	2	TOENA	パルス幅測定モードでは“0”にします。	
	1	TOPCR		
	0	TEDGSEL	H レベル幅を測定するために“1”にします。	
TRAMR	7	TCKCUT	カウントソースを供給するため“0”にします。	0x13
	6	TCK2	カウントソースを f8 にするため“001”にします。	
	5	TCK1		
	4	TCK0		
	3	-	何も配置されていないので、“0”にします。	
	2	TMOD2	パルス幅測定モードにするため“011”にします。	
	1	TMOD1		
	0	TMOD0		
TRAPRE			カウントダウンのため“0xff”にします。	0xff
TRA			カウントダウンのため“0xff”にします。	0xff
TRASR	7	-	何も配置されていないので、“0”にします。	0x03
	6	-		
	5	-		
	4	TRAOSEL1	使用しません。“0”にしておきます。	
	3	TRAOSEL0		
	2	-	何も配置されていないので、“0”にします。	
	1	TRAIOSEL1	TRAIO 端子を P3_2 に割り当てるので、“11”にします。	
	0	TRAIOSEL0		

レジスタ	ビット	シンボル	説明	設定値
TRAIC	7	-	何も配置されていないので、“0”にします。	0x01
	6	-		
	5	-		
	4	-		
	3	IR	割り込み要求ビット	
	2	ILVL2	割り込みレベルを 1 にするため、“001”にします。	
	1	ILVL1		
	0	ILVL0		
TRACR	7	-	何も配置されていないので、“0”にします。	0x01
	6	-		
	5	TUNDF	アンダーフローフラグ	
	4	TEDGF	有効エッジ判定フラグ	
	3	-	何も配置されていないので、“0”にします。	
	2	TSTOP	カウントを強制停止しないので“0”にします。	
	1	TCSTF	タイマーRA カウントステータスフラグ	
	0	TSTART	カウントを開始するため、“1”にします。	

3.6　割り込みプログラム：intTRAIC 関数

intTRAIC 関数はパルス幅測定モードでアンダーフロー時もしくは、TRAIO 端子の入力の立ち下り時に割り込みで実行されます。

プログラム

```
289 :  #pragma interrupt intTRAIC (vect=22)
290 :  void intTRAIC( void )
291 :  {
292 :          static int      num = -1;
293 :          unsigned short  data;
294 :          int             i;
295 :
296 :          // アンダーフロー
297～307 :  「3.6.1 アンダーフロー」を参照。
308 :
309 :          // エッジ検出
310～349 :  「3.6.2 エッジ検出」を参照。
350 :
351 :  }
```

3.6.1　アンダーフロー

プログラム

```
297 :          if( tundf_tracr == 1){
300 :        「(1) アンダーフローフラグのクリア」を参照。
302～305 :  「(2) タイマーカウンタのクリア」を参照。
307 :          }
```

赤外線の入力がない場合、エッジ検出がされず、タイマーカウンタのアンダーフローがおこりますので、アンダーフローフラグのクリアとタイマーカウンタのクリアをおこないます。

(1) アンダーフローフラグのクリア

```
300 :                  tundf_tracr = 0;
```

TRACR レジスタの TUNDF ビットをクリアします。

(2) タイマーカウンタのクリア

```
302 :                  tstart_tracr = 0;       // カウントストップ
303 :                  tra = 0xff;             // カウンタをリセット
304 :                  trapre = 0xff;          // カウンタをリセット
305 :                  tstart_tracr = 1;       // カウントスタート
```

タイマーのカウントをストップさせて、TRA レジスタと TRAPRE レジスタに 0xff をセットします。セット後に、カウントをスタートします。

3.6.2　エッジ検出

プログラム

```
310 :          if( tedgf_tracr == 1){
313 :        「(1) エッジ検出フラグのクリア」を参照。
316～319 :  「(2) カウンタ値の取得」を参照。
322～325 :  「(3) リーダーコード検出 (1 回目)」を参照。
328～331 :  「(4) 0 受信」を参照。
334～337 :  「(5) 1 受信」を参照。
340～342 :  「(6) リーダーコード検出 (リピート)」を参照。
340～342 :  「(7) タイマーカウンタのクリア」を参照。
349 :          }
```

(1) エッジ検出フラグのクリア

```
313 :                  tedgf_tracr = 0;
```

TRACR レジスタの TEDGF ビットをクリアします。

(2) カウンタ値の取得

```
316 :                  data = tra;
317 :                  data = data << 8;
318 :                  data = data + trapre;
319 :                  data = 65535 - data;
```

TRA レジスタと TRAPRE レジスタで 16 ビットのカウンタとなりますので、それぞれの値を合計して、16 ビットの最大値の 65535 から引くことによって、受信したパルス幅がわかります。

(3) リーダーコード検出 (1 回目)

```
322 :                  if( ( DefIrS - DefIrM ) < data && data < ( DefIrS + DefIrM ) ){
323 :                          num = 0;
324 :                          ir_data[0] = 'S';
325 :                  }
```

シンボル定義で指定したリーダーコード (1 回目) ±マージンの範囲に、受信したパルス幅が入っていた場合、num 変数を“0”にして、ir_data 配列の先頭に“S”を入れます。

(4) 0 受信

```
328 :                  if( ( DefIr0 - DefIrM ) < data && data < ( DefIr0 + DefIrM ) && 0 <= num && num <= 31 ){
329 :                          num++;
330 :                          ir_data[num] = '0';
331 :                  }
```

シンボル定義で指定した 0 の幅±マージンの範囲に、受信したパルス幅が入っていた場合、num 変数をインクリメントして、ir_data 配列に“0”を入れます。

(5) 1 受信

```
334 :                  if( ( DefIr1 - DefIrM ) < data && data < ( DefIr1 + DefIrM ) && 0 <= num && num <= 31 ){
335 :                          num++;
336 :                          ir_data[num] = '1';
337 :                  }
```

シンボル定義で指定した 1 の幅±マージンの範囲に、受信したパルス幅が入っていた場合、num 変数をインクリメントして、ir_data 配列に“1”を入れます。

（6）リーダーコード検出（リピート）

```
340 :                   if( ( DefIrR - DefIrM ) < data && data < ( DefIrR + DefIrM ) && num == 32 ){
341 :                           ir_data[0] = 'R';
342 :                   }
```

シンボル定義で指定したリーダーコード（リピート）±マージンの範囲に、受信したパルス幅が入っていた場合、ir_data 配列の先頭に“R”を入れます。

（7）タイマーカウンタクリア

```
344 :                   tstart_tracr = 0;       // カウントストップ
345 :                   tra = 0xff;             // カウンタをリセット
346 :                   trapre = 0xff;          // カウンタをリセット
347 :                   tstart_tracr = 1;       // カウントスタート
```

タイマーのカウントをストップさせて、TRA レジスタと TRAPRE レジスタに 0xff をセットします。セット後に、カウントをスタートします。

3.7　メインプログラム：main 関数

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

プログラム

```
 64 :  void main(void)
 65 :  {
 66 :          int i;
 67 :          int button;
 68 :
 69 :          // 初期化
 70 :          init();
 71 :          init_uart0_printf(SPEED_9600);
 72 :
 73 :          while(1){
 75～ 87 :  「3.7.1 データ受信チェック」を参照。
 89～135 :  「3.7.2 データチェック」を参照。
137～166 :  「3.7.3 モーター駆動」を参照。
168～173 :  「3.7.4 デバッグ出力」を参照。
175 :          }
176 :
177 :  }
```

main 関数では、割り込みで受信した赤外線のデータを判断して、モーターを制御しています。

3.7.1　データ受信チェック

```
 75 :                  // 赤外線の受信データのリーダーコード、リピートリーダーコードをクリア
 76 :                  ir_data[0] = ' ';
 77 :
 78 :                  // 受信のため待ち
 79 :                  timer(200);
 80 :
 81 :                  // リーダーコード、リピートリーダーコードが受信されていなかった場合
 82 :                  if( ir_data[0] == ' ' ){
 83 :                          // データをすべてクリア
 84 :                          for(i=0;i<60;i++){
 85 :                                  ir_data[i] = ' ';
 86 :                          }
 87 :                  }
```

ir_data 配列の先頭をクリアし、200「ms」待ちます。200「ms」待っている間に割り込みで赤外線データを受信した場合、先頭に“S”か“R”が入ってきますが、入らなかった場合は、データをすべてクリアして、次の受信に備えます。

3.7.2　データチェック

```
 89 :                  // ボタンが押されていない
 90 :                  button = 0;
 91 :  #if 1
 92 :  //139 日立 地上アナログテレビ
 93 :                  // 2 ボタン
 94 :                  if( strncmp( &ir_data[0], "S0000101011110101011100001000111", 1+32 ) == 0 ||
 95 :                          strncmp( &ir_data[0], "R0000101011110101011100001000111", 1+32 ) == 0  ){
 96 :                          button = 2;
 97 :                  }
 98 :                  // 4 ボタン
 99 :                  if( strncmp( &ir_data[0], "S0000101011110101001110001100011", 1+32 ) == 0 ||
100 :                          strncmp( &ir_data[0], "R0000101011110101001110001100011", 1+32 ) == 0  ){
101 :                          button = 4;
102 :                  }
103 :                  // 6 ボタン
104 :                  if( strncmp( &ir_data[0], "S0000101011110101011110001000011", 1+32 ) == 0 ||
105 :                          strncmp( &ir_data[0], "R0000101011110101011110001000011", 1+32 ) == 0  ){
106 :                          button = 6;
107 :                  }
108 :                  // 8 ボタン
109 :                  if( strncmp( &ir_data[0], "S0000101011110101001000001101111", 1+32 ) == 0 ||
110 :                          strncmp( &ir_data[0], "R0000101011110101001000001101111", 1+32 ) == 0  ){
111 :                          button = 8;
112 :                  }
113 :  #else
114 :  //134 NEC 地上アナログテレビ
115 :                  // 2 ボタン
116 :                  if( strncmp( &ir_data[0], "S0001100011100111100010000111011", 1+32 ) == 0 ||
117 :                          strncmp( &ir_data[0], "R0001100011100111100010000111011", 1+32 ) == 0 ){
118 :                          button = 2;
119 :                  }
120 :                  // 4 ボタン
121 :                  if( strncmp( &ir_data[0], "S0001100011100111110010000011011", 1+32 ) == 0 ||
122 :                          strncmp( &ir_data[0], "R0001100011100111110010000011011", 1+32 ) == 0 ){
123 :                          button = 4;
124 :                  }
125 :                  // 6 ボタン
126 :                  if( strncmp( &ir_data[0], "S0001100011100111101010000101011", 1+32 ) == 0 ||
127 :                          strncmp( &ir_data[0], "R0001100011100111101010000101011", 1+32 ) == 0 ){
128 :                          button = 6;
129 :                  }
130 :                  // 8 ボタン
131 :                  if( strncmp( &ir_data[0], "S0001100011100111111010000001011", 1+32 ) == 0 ||
132 :                          strncmp( &ir_data[0], "R0001100011100111111010000001011", 1+32 ) == 0 ){
133 :                          button = 8;
134 :                  }
135 :  #endif
```

ir_data 配列に格納された受信データを比較して、どのボタンが押されたかを button 変数に格納します。何も押されていない場合は“0”になります。

3.7.3　モーター駆動

```
137 :                   switch( button ){
 138 :                   // ボタンが押されていないか
 139 :                   case 0:
 140 :                           motor( 0, 0 );
 141 :                           break;
 142 :
 143 :                   // 2 ボタンで前進
 144 :                   case 2:
 145 :                           motor( 100, 100 );
 146 :                           break;
 147 :
 148 :                   // 4 ボタンで左旋回
 149 :                   case 4:
 150 :                           motor( 0, 100 );
 151 :                           break;
 152 :
 153 :                   // 6 ボタンで右旋回
 154 :                   case 6:
 155 :                           motor( 100, 0 );
 156 :                           break;
 157 :
 158 :                   // 8 ボタンでバック
 159 :                   case 8:
 160 :                           motor( -100, -100 );
 161 :                           break;
 162 :
 163 :                   default:
 164 :                           break;
 165 :
 166 :                   }
```

button 変数に格納されたデータによってミニマイコンカーVer.2 の移動する方向を決めます。

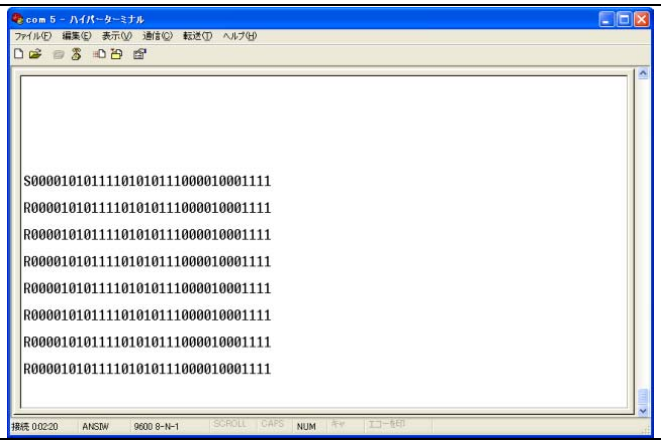
3.7.4　デバッグ出力

```
 168 :                   // デバッグ出力
 169 :                   for(i=0;i<60;i++){
 170 :                           printf("%c",ir_data[i]);
 171 :                   }
 172 :                   printf("¥n");
 173 :                   printf("¥n");
```

ir_data 配列の中身をシリアルで出力します。

3.8　赤外線リモコンのデータ解析

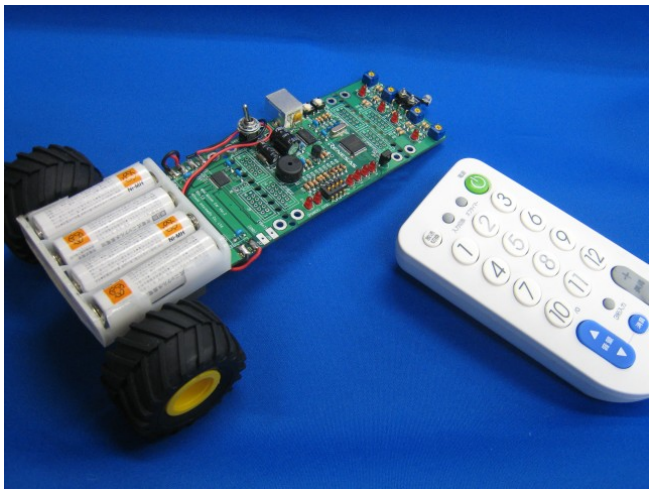
受信データがどのようになっているかを見る場合は、ミニマイコンカーVer.2 の電源を入れた状態で、ミニマイコンカーVer.2 と PC を USB ケーブルで接続し、ハイパーターミナルなどの通信ソフトで確認をおこなうことができます。



ボタンを押すと、受信データが表示されます（。受信データは、“S”もしくは“R”と 32 個の数字で構成されています（タイミングによってすべて表示されない場合があります)。

```
94 :                    if( strncmp( &ir_data[0], "S00001010111101010111000010001111", 1+32 ) == 0 ||
95 :                            strncmp( &ir_data[0], "R00001010111101010111000010001111", 1+32 ) == 0  ){
96 :                            button = 2;
97 :                    }
```

表示された数字部分をコピーして、ir_data 配列と文字列を比較している部分に入れることで、リモコンのボタンに対応させることができます。



改定記録	赤外線受光モジュール　C言語リモコンプログラム 解説マニュアル

版	発行日	改定内容	
		ページ	ポイント

赤外線受光モジュール　C 言語リモコンプログラム解説マニュアル

発行年月日　　2010 年 4 月 26 日　　第 1.00 版

発行　　日立インターメディックス株式会社
〒101-0054　東京都千代田区神田錦町 2 丁目 1 番地 5