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

第 1.05 版
2015 年 7 月 15 日
株式会社日立ドキュメントソリューションズ

5. 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ただけでは、クランクやレーンチェンジを正常に曲がれません。motor 関数を呼び出すときの引数を調整して、正常に曲がれるようにしましょう。DIP スイッチの設定は、以下のようにしてください。

DIP スイッチ（ON：0、OFF：1）			
P5_7 (3)	P4_5 (2)	P4_4 (1)	P4_3 (0)
0	0	0	1

5.1 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     走行プログラム
 4 :  // バージョン   Ver.1.00
 5 :  // Date         2009.07.01
 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 "sfr_r835a.h"
13 :
14 :  //---------------------------------------------------------------------------
15 :  // シンボル定義
16 :  //---------------------------------------------------------------------------
17 :  #define TIMER_CYCLE     155             // 1ms:0.001/(1/(20000000/128))-1
18 :  #define PWM_CYCLE       39999           // 16ms:0.016/(1/(20000000/8))-1
19 :
20 :  #define Def_500Hz       4999            // 500Hz:(1/500)/(1/(20000000/8))-1
21 :  #define Def_1000Hz      2499            // 1000Hz:(1/1000)/(1/(20000000/8))-1
22 :
23 :  #define Def_C3          19083           // ド:(1/131)/(1/(20000000/8))-1
24 :  #define Def_D3          17006           // レ:(1/147)/(1/(20000000/8))-1
25 :  #define Def_E3          15151           // ミ:(1/165)/(1/(20000000/8))-1
26 :  #define Def_F3          14285           // ファ:(1/175)/(1/(20000000/8))-1
27 :  #define Def_G3          12754           // ソ:(1/196)/(1/(20000000/8))-1
28 :  #define Def_A3          11362           // ラ:(1/220)/(1/(20000000/8))-1
29 :  #define Def_B3          10120           // シ:(1/247)/(1/(20000000/8))-1
30 :  #define Def_C4          9541            // ド:(1/262)/(1/(20000000/8))-1
31 :
32 :  #define DI()            asm("FCLR I")   // 割り込み禁止
33 :  #define EI()            asm("FSET I")   // 割り込み許可
34 :
35 :  //---------------------------------------------------------------------------
36 :  // 関数プロトタイプの宣言
37 :  //---------------------------------------------------------------------------
38 :  void init( void );
39 :  unsigned char sensor( void );
40 :  void motor( int data1, int data2 );
```

```
 41 :  void timer( unsigned long timer_set );
 42 :  void beep( int data1 );
 43 :  unsigned char dipsw( void );
 44 :  unsigned char pushsw( void );
 45 :
 46 :  //------------------------------------------------------------------------------
 47 :  // グローバル変数の宣言
 48 :  //------------------------------------------------------------------------------
 49 :  unsigned long   cnt0 = 0;              // timer 関数用
 50 :  unsigned long   cnt1 = 0;              // main 内で使用
 51 :  int             pattern = 0;           // パターン番号
 52 :
 53 :  //------------------------------------------------------------------------------
 54 :  // メインプログラム
 55 :  //------------------------------------------------------------------------------
 56 :  void main(void)
 57 :  {
 58 :          // 初期化
 59 :          init();
 60 :
 61 :          // 起動音
 62 :          beep(Def_500Hz);
 63 :          timer(100);
 64 :          beep(Def_1000Hz);
 65 :          timer(100);
 66 :          beep(0);
 67 :
 68 :          while(1){
 69 :          switch( pattern ){
 70 :
 71 :          //-----------------------------------------------------------------
 72 :          // パターンについて
 73 :          //  0 : スイッチ入力待ち
 74 :          //  1 : 1 秒後にスタート
 75 :          // 11 : 通常トレース
 76 :          // 21 : クロスライン検出後のトレース、クランク検出
 77 :          // 22 : クランクの曲げ動作継続処理
 78 :          // 31 : 左ハーフライン検出後のトレース、左レーンチェンジ検出
 79 :          // 32 : 左レーンチェンジ曲げ動作継続処理
 80 :          // 33 : 左レーンチェンジ終了検出
 81 :          // 41 : 右ハーフライン検出後のトレース、右レーンチェンジ検出
 82 :          // 42 : 右レーンチェンジ曲げ動作継続処理
 83 :          // 43 : 右レーンチェンジ終了検出
 84 :          //-----------------------------------------------------------------
 85 :
 86 :          case 0:
 87 :                  // スイッチ入力待ち
 88 :                  if( pushsw() == 1 ){
 89 :                          beep(Def_1000Hz);
 90 :                          cnt1 = 0;
 91 :                          pattern = 1;
 92 :                  }
 93 :
 94 :                  break;
 95 :
 96 :          case 1:
 97 :                  // 1 秒後にスタート
 98 :                  if( cnt1 >= 1000 ){
 99 :                          beep(0);
100 :                          cnt1 = 0;
101 :                          pattern = 11;
102 :                  }
103 :
```

```
104 :                   break;
105 :
106 :           case 11:
107 :                   // 通常トレース
108 :                   beep(0);
109 :
110 :                   switch( ( sensor() & 0x0f ) ){
111 :                   case 0x06:
112 :                           // 0000 0110 センター→まっすぐ
113 :                           motor( 100, 100 );
114 :                           break;
115 :
116 :                   case 0x04:
117 :                           // 0000 0100 少し右寄り→左へ小曲げ
118 :                           motor( 85, 100 );
119 :                           break;
120 :
121 :                   case 0x0c:
122 :                           // 0000 1100 中くらい右寄り→左へ中曲げ
123 :                           motor( 70, 100 );
124 :                           break;
125 :
126 :                   case 0x08:
127 :                           // 0000 1000 大きく右寄り→左へ大曲げ
128 :                           motor( 55, 100 );
129 :                           break;
130 :
131 :                   case 0x02:
132 :                           // 0000 0010 少し左寄り→右へ小曲げ
133 :                           motor( 100, 85 );
134 :                           break;
135 :
136 :                   case 0x03:
137 :                           // 0000 0011 中くらい左寄り→右へ中曲げ
138 :                           motor( 100, 70 );
139 :                           break;
140 :
141 :                   case 0x01:
142 :                           // 0000 0001 大きく左寄り→右へ大曲げ
143 :                           motor( 100, 55 );
144 :                           break;
145 :
146 :                   case 0x0f:
147 :                           // 0000 1111 クロスライン検出
148 :                           motor( 100, 100 );
149 :                           cnt1 = 0;
150 :                           pattern = 21;
151 :                           break;
152 :
153 :                   case 0x0e:
154 :                           // 0000 1110 左ハーフライン検出
155 :                           motor( 100, 100 );
156 :                           cnt1 = 0;
157 :                           pattern = 31;
158 :                           break;
159 :
160 :                   case 0x07:
161 :                           // 0000 0111 右ハーフライン検出
162 :                           motor( 100, 100 );
163 :                           cnt1 = 0;
164 :                           pattern = 41;
165 :                           break;
166 :
```

```
167 :                   default:
168 :                           break;
169 :
170 :                   }
171 :
172 :                   break;
173 :
174 :           case 21:
175 :                   // クロスライン検出後のトレース、クランク検出
176 :                   beep(Def_C3);
177 :
178 :                   switch( ( sensor() & 0x0f ) ){
179 :                   case 0x06:
180 :                           // 0000 0110 センタ→まっすぐ
181 :                           motor( 100, 100 );
182 :                           break;
183 :
184 :                   case 0x04:
185 :                           // 0000 0100 少し右寄り→左へ小曲げ
186 :                           motor( 85, 100 );
187 :                           break;
188 :
189 :                   case 0x0c:
190 :                           // 0000 1100 中くらい右寄り→左へ中曲げ
191 :                           motor( 70, 100 );
192 :                           break;
193 :
194 :                   case 0x08:
195 :                           // 0000 1000 大きく右寄り→左へ大曲げ
196 :                           motor( 55, 100 );
197 :                           break;
198 :
199 :                   case 0x02:
200 :                           // 0000 0010 少し左寄り→右へ小曲げ
201 :                           motor( 100, 85 );
202 :                           break;
203 :
204 :                   case 0x03:
205 :                           // 0000 0011 中くらい左寄り→右へ中曲げ
206 :                           motor( 100, 70 );
207 :                           break;
208 :
209 :                   case 0x01:
210 :                           // 0000 0001 大きく左寄り→右へ大曲げ
211 :                           motor( 100, 55 );
212 :                           break;
213 :
214 :                   default:
215 :                           break;
216 :
217 :                   }
218 :
219 :                   if( cnt1 >= 1000 ){
220 :                           switch( ( sensor() & 0x0f ) ){
221 :                           case 0x0e:
222 :                                   // 0000 1110 左クランク検出
223 :                                   motor( 0, 90 );
224 :                                   cnt1 = 0;
225 :                                   pattern = 22;
226 :                                   break;
227 :
228 :                           case 0x07:
229 :                                   // 0000 0111 右クランク検出
```

```
230 :                                   motor( 90, 0 );
231 :                                   cnt1 = 0;
232 :                                   pattern = 22;
233 :                                   break;
234 :
235 :                           default:
236 :                                   break;
237 :
238 :                           }
239 :                   }
240 :
241 :                   break;
242 :
243 :           case 22:
244 :                   // クランクの曲げ動作継続処理
245 :                   if( cnt1 >= 1000 ){
246 :                           pattern = 11;
247 :                   }
248 :
249 :                   break;
250 :
251 :           case 31:
252 :                   // 左ハーフライン検出後のトレース、左レーンチェンジ検出
253 :                   beep(Def_D3);
254 :
255 :                   switch( ( sensor() & 0x0f ) ){
256 :                   case 0x06:
257 :                           // 0000 0110 センタ→まっすぐ
258 :                           motor( 100, 100 );
259 :                           break;
260 :
261 :                   case 0x04:
262 :                           // 0000 0100 少し右寄り→左へ小曲げ
263 :                           motor( 85, 100 );
264 :                           break;
265 :
266 :                   case 0x0c:
267 :                           // 0000 1100 中くらい右寄り→左へ中曲げ
268 :                           motor( 70, 100 );
269 :                           break;
270 :
271 :                   case 0x08:
272 :                           // 0000 1000 大きく右寄り→左へ大曲げ
273 :                           motor( 55, 100 );
274 :                           break;
275 :
276 :                   case 0x02:
277 :                           // 0000 0010 少し左寄り→右へ小曲げ
278 :                           motor( 100, 85 );
279 :                           break;
280 :
281 :                   case 0x03:
282 :                           // 0000 0011 中くらい左寄り→右へ中曲げ
283 :                           motor( 100, 70 );
284 :                           break;
285 :
286 :                   case 0x01:
287 :                           // 0000 0001 大きく左寄り→右へ大曲げ
288 :                           motor( 100, 55 );
289 :                           break;
290 :
291 :                   case 0x0f:
292 :                           // 0000 1111 クロスライン検出
```

```
293 :                           motor( 100, 100 );
294 :                           cnt1 = 0;
295 :                           pattern = 21;
296 :                           break;
297 :
298 :                   default:
299 :                           break;
300 :
301 :                   }
302 :
303 :                   if( cnt1 >= 1000 ){
304 :                           switch( ( sensor() & 0x0f ) ){
305 :                           case 0x00:
306 :                                   // 0000 0000 左レーンチェンジ検出
307 :                                   motor( 0, 100 );
308 :                                   cnt1 = 0;
309 :                                   pattern = 32;
310 :                                   break;
311 :
312 :                           default:
313 :                                   break;
314 :
315 :                           }
316 :                   }
317 :
318 :                   break;
319 :
320 :           case 32:
321 :                   // 左レーンチェンジ曲げ動作継続処理
322 :                   if( cnt1 >= 700 ){
323 :                           motor( 100, 100 );
324 :                           cnt1 = 0;
325 :                           pattern = 33;
326 :                   }
327 :
328 :                   break;
329 :
330 :           case 33:
331 :                   // 左レーンチェンジ終了検出
332 :                   if( cnt1 >= 500 ){
333 :                           switch( ( sensor() & 0x0f ) ){
334 :                           case 0x01:
335 :                                   // 0000 0001 左レーンチェンジ終了検出
336 :                                   pattern = 11;
337 :                                   break;
338 :                           default:
339 :                                   break;
340 :                           }
341 :                   }
342 :
343 :                   break;
344 :
345 :           case 41:
346 :                   // 右ハーフライン検出後のトレース、右レーンチェンジ検出
347 :                   beep(Def_E3);
348 :
349 :                   switch( ( sensor() & 0x0f ) ){
350 :                   case 0x06:
351 :                           // 0000 0110 センター→まっすぐ
352 :                           motor( 100, 100 );
353 :                           break;
354 :
355 :                   case 0x04:
```

```
356 :                           // 0000 0100 少し右寄り→左へ小曲げ
357 :                           motor( 85, 100 );
358 :                           break;
359 :
360 :                   case 0x0c:
361 :                           // 0000 1100 中くらい右寄り→左へ中曲げ
362 :                           motor( 70, 100 );
363 :                           break;
364 :
365 :                   case 0x08:
366 :                           // 0000 1000 大きく右寄り→左へ大曲げ
367 :                           motor( 55, 100 );
368 :                           break;
369 :
370 :                   case 0x02:
371 :                           // 0000 0010 少し左寄り→右へ小曲げ
372 :                           motor( 100, 85 );
373 :                           break;
374 :
375 :                   case 0x03:
376 :                           // 0000 0011 中くらい左寄り→右へ中曲げ
377 :                           motor( 100, 70 );
378 :                           break;
379 :
380 :                   case 0x01:
381 :                           // 0000 0001 大きく左寄り→右へ大曲げ
382 :                           motor( 100, 55 );
383 :                           break;
384 :
385 :                   case 0x0f:
386 :                           // 0000 1111 クロスライン検出
387 :                           motor( 100, 100 );
388 :                           cnt1 = 0;
389 :                           pattern = 21;
390 :                           break;
391 :
392 :                   default:
393 :                           break;
394 :
395 :                   }
396 :
397 :                   if( cnt1 >= 1000 ){
398 :                           switch( ( sensor() & 0x0f ) ){
399 :                           case 0x00:
400 :                                   // 0000 0000 右レーンチェンジ検出
401 :                                   motor( 100, 0 );
402 :                                   cnt1 = 0;
403 :                                   pattern = 42;
404 :                                   break;
405 :
406 :                           default:
407 :                                   break;
408 :
409 :                           }
410 :                   }
411 :
412 :                   break;
413 :
414 :           case 42:
415 :                   // 右レーンチェンジ曲げ動作継続処理
416 :                   if( cnt1 >= 700 ){
417 :                           motor( 100, 100 );
418 :                           cnt1 = 0;
```

```
419 :                           pattern = 43;
420 :                   }
421 :
422 :                   break;
423 :
424 :           case 43:
425 :                   // 右レーンチェンジ終了検出
426 :                   if( cnt1 >= 500 ){
427 :                           switch( ( sensor() & 0x0f ) ){
428 :                           case 0x08:
429 :                                   // 0000 1000 右レーンチェンジ終了検出
430 :                                   pattern = 11;
431 :                                   break;
432 :                           default:
433 :                                   break;
434 :                           }
435 :                   }
436 :
437 :                   break;
438 :
439 :           default:
440 :                   break;
441 :
442 :           }
443 :           }
444 : }
445 :
446 : //--------------------------------------------------------------------------------
447 : // R8C/35A の内蔵周辺機能の初期化
448 : //--------------------------------------------------------------------------------
449 : void init( void )
450 : {
451 :           unsigned char i = 0;
452 :
453 :           // 割り込み禁止
454 :           DI();
455 :
456 :           // クロック発生回路の XIN クロック設定
457 :           prc0 = 1;
458 :
459 :           cm13 = 1;
460 :           cm05 = 0;
461 :           while(i <= 50) i++;
462 :           ocd2 = 0;
463 :
464 :           prc0 = 0;
465 :
466 :           // I/O ポートの入出力設定
467 :           prc2 = 1;                       // pd0 レジスタへの書き込み許可
468 :           pd0 = 0xe0;                     // P0_0～P0_3:センサー
469 :                                           // P0_4:マイクロスイッチ
470 :                                           // P0_5～P0_7:LED
471 :           prc2 = 0;                       // pd0 レジスタへの書き込み禁止
472 :           pd1 = 0xdf;                     // P1_0～P1_3:LED
473 :                                           // P1_4:TXD0
474 :                                           // P1_5:RXD0
475 :           pd2 = 0xfe;                     // P2_0:スイッチ
476 :                                           // P2_1:AIN1
477 :                                           // P2_2:PWMA
478 :                                           // P2_3:BIN1
479 :                                           // P2_4:PWMB
480 :                                           // P2_5:SERVO
481 :                                           // P2_6:AIN2
```

```
482 :                                           // P2_7:BIN2
483 :           pd3 = 0xfb;                     // P3_2:赤外線受信
484 :                                           // P3_4:ブザー
485 :           pd4 = 0x80;                     // P4_2:VREF
486 :                                           // P4_3～P4_5:DIPSW
487 :                                           // P4_6:XIN
488 :                                           // P4_7:XOUT
489 :           pd5 = 0x40;                     // P5_7:DIPSW
490 :           pd6 = 0xff;                     //
491 :
492 :
493 :
494 :           mstcr = 0x00;                   // モジュールストップ解除
495 :
496 :
497 :
498 :           // タイマ RB の 1ms 割り込み設定
499 :           trbmr = 0x00;                   // カウントソースは f1
500 :           trbpre = 128 - 1;               // プリスケーラ
501 :           trbpr = TIMER_CYCLE;            // プライマリカウンタ
502 :           trbic = 0x01;                   // タイマ RB の割り込みレベル設定
503 :           trbcr = 0x01;                   // カウントを開始
504 :
505 :           // タイマ RC の PWM モード
506 :           trccr1 = 0xb0;                  // カウントソースは f8
507 :           trcgra = 0;                     // 圧電サウンダの周期
508 :           trcgrc = 0;                     // 圧電サウンダのデューティ比
509 :           trccr2 = 0x02;                  // TRCIOC 端子はアクティブレベル H
510 :           trcoer = 0x0b;                  // TRCIOC 端子の出力許可
511 :           trcpsr1 = 0x02;                 // TRCIOC 端子を P3_4 に割り当て
512 :           trcmr = 0x8a;                   // カウントを開始
513 :
514 :           // タイマ RD のリセット同期 PWM モード
515 :           trdpsr0 = 0x08;                 // TRDIOB0 端子を P2_2 に割り当て
516 :           trdpsr1 = 0x05;                 // TRDIOB1 端子を P2_5 に割り当て
517 :                                           // TRDIOA1 端子を P2_4 に割り当て
518 :           trdmr = 0xf0;                   // レジスタをバッファ動作にする
519 :           trdfcr = 0x01;                  // リセット同期 PWM モードに設定
520 :           trdoer1 = 0xcd;                 // TRDIOB1 の出力許可
521 :                                           // TRDIOA1 の出力許可
522 :                                           // TRDIOB0 端子の出力許可
523 :           trdcr0 = 0x23;                  // カウントソースは f8
524 :           trdgra0 = trdgrc0 = PWM_CYCLE;  // 周期
525 :           trdgrb0 = trdgrd0 = 0;          // TRDIOB0 端子 (左モータ)
526 :           trdgra1 = trdgrc1 = 0;          // TRDIOA1 端子 (右モータ)
527 :           trdgrb1 = trdgrd1 = 0;          // TRDIOB1 端子 (サーボ)
528 :           trdstr = 0x0d;                  // カウントを開始
529 :
530 :           // 割り込み許可
531 :           EI();
532 :   }
533 :
534 :   //------------------------------------------------------------------------------
535 :   // 割り込み
536 :   //------------------------------------------------------------------------------
537 :   #pragma interrupt intTRBIC (vect=24)
538 :   void intTRBIC( void )
539 :   {
540 :           p0_7 = ~p0_7;
541 :
542 :           if( p0_7 == 0 ){
543 :                   //p0_1、p0_3 のモニタが可能
544 :                   p0_5 = ~p0_1;
```

```
545 :                   p0_6 = ~p0_3;
546 :           }else{
547 :                   //p0_0、p0_2 のモニタが可能
548 :                   p0_5 = p0_0;
549 :                   p0_6 = p0_2;
550 :           }
551 :
552 :           cnt0++;
553 :           cnt1++;
554 :   }
555 :
556 :   //--------------------------------------------------------------------------------
557 :   // センサー状態検出
558 :   // 引数         なし
559 :   // 戻り値       センサ値
560 :   //--------------------------------------------------------------------------------
561 :   unsigned char sensor( void )
562 :   {
563 :           volatile unsigned char  data1;
564 :
565 :           data1 = ~p0;                    // ラインの色は白
566 :           data1 = data1 & 0x0f;
567 :
568 :           return( data1 );
569 :   }
570 :
571 :   //--------------------------------------------------------------------------------
572 :   // モーター速度制御
573 :   // 引数         左モータ:-100～100、右モータ:-100～100
574 :   //              0 で停止、100 で正転 100%、-100 で逆転 100%
575 :   // 戻り値       なし
576 :   //--------------------------------------------------------------------------------
577 :   void motor( int data1, int data2 )
578 :   {
579 :           volatile int    motor_r;
580 :           volatile int    motor_l;
581 :           volatile int    sw_data;
582 :
583 :           sw_data = dipsw() + 5;
584 :           motor_l = (long)data1 * sw_data / 20;
585 :           motor_r = (long)data2 * sw_data / 20;
586 :
587 :           if( motor_l >= 0 ) {
588 :                   p2_1 = 0;
589 :                   p2_6 = 1;
590 :                   trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
591 :           } else {
592 :                   p2_1 = 1;
593 :                   p2_6 = 0;
594 :                   trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
595 :           }
596 :
597 :           if( motor_r >= 0 ) {
598 :                   p2_3 = 0;
599 :                   p2_7 = 1;
600 :                   trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
601 :           } else {
602 :                   p2_3 = 1;
603 :                   p2_7 = 0;
604 :                   trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
605 :           }
606 :   }
607 :
```

```
608 :  //-----------------------------------------------------------------------------
609 :  // 時間稼ぎ
610 :  // 引数         タイマ値 1=1ms
611 :  // 戻り値       なし
612 :  //-----------------------------------------------------------------------------
613 :  void timer( unsigned long data1 )
614 :  {
615 :          cnt0 = 0;
616 :          while( cnt0 < data1 );
617 :  }
618 :
619 :  //-----------------------------------------------------------------------------
620 :  // 音を鳴らす
621 :  // 引数         (1/音の周波数)/(1/(クロック周波数/8))-1
622 :  // 戻り値       なし
623 :  //-----------------------------------------------------------------------------
624 :  void beep( int data1 )
625 :  {
626 :          trcgra = data1;                 // 周期の設定
627 :          trcgrc = data1 / 2;             // デューティ 50%のため周期の半分の値
628 :  }
629 :
630 :  //-----------------------------------------------------------------------------
631 :  // DIP スイッチ状態検出
632 :  // 引数         なし
633 :  // 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。
634 :  //-----------------------------------------------------------------------------
635 :  unsigned char dipsw( void )
636 :  {
637 :          volatile unsigned char  data1;
638 :
639 :          data1 = ( ( p5 >> 4 ) & 0x08 ) | ( ( p4 >> 3 ) & 0x07 );
640 :
641 :          return( data1 );
642 :  }
643 :
644 :  //-----------------------------------------------------------------------------
645 :  // プッシュスイッチ状態検出
646 :  // 引数         なし
647 :  // 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
648 :  //-----------------------------------------------------------------------------
649 :  unsigned char pushsw( void )
650 :  {
651 :          unsigned char data1;
652 :
653 :          data1 = ~p2;
654 :          data1 &= 0x01;
655 :
656 :          return( data1 );
657 :  }
```

5.2 スタート

プログラム

```
 1 :  //------------------------------------------------------------------------------
 2 :  // 対象マイコン R8C/35A
 3 :  // ﾌｧｲﾙ内容     走行プログラム
 4 :  // バージョン   Ver.1.00
 5 :  // Date         2009.07.01
 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 :  //------------------------------------------------------------------------------
```

最初はコメント部分です。「//」の後の文字はコンパイル時に無視されるので、コメントを書くときに利用します。

5.3 外部ファイルの読み込み（インクルード）

プログラム

```
12 :  #include "sfr_r835a.h"
```

#include は外部のファイルを読み込むときに使用します。

名称	説明
sfr_r835a.h	R8C/35A 用の内蔵周辺機能の制御レジスタを定義したファイルです。

5.4 シンボル定義

プログラム

```
17 :  #define TIMER_CYCLE     155           // 1ms:0.001/(1/(20000000/128))-1
18 :  #define PWM_CYCLE       39999         // 16ms:0.016/(1/(20000000/8))-1
19 :
20 :  #define Def_500Hz       4999          // 500Hz:(1/500)/(1/(20000000/8))-1
21 :  #define Def_1000Hz      2499          // 1000Hz:(1/1000)/(1/(20000000/8))-1
22 :
23 :  #define Def_C3          19083         // ド:(1/131)/(1/(20000000/8))-1
24 :  #define Def_D3          17006         // レ:(1/147)/(1/(20000000/8))-1
25 :  #define Def_E3          15151         // ミ:(1/165)/(1/(20000000/8))-1
26 :  #define Def_F3          14285         // ファ:(1/175)/(1/(20000000/8))-1
27 :  #define Def_G3          12754         // ソ:(1/196)/(1/(20000000/8))-1
28 :  #define Def_A3          11362         // ラ:(1/220)/(1/(20000000/8))-1
29 :  #define Def_B3          10120         // シ:(1/247)/(1/(20000000/8))-1
30 :  #define Def_C4          9541          // ド:(1/262)/(1/(20000000/8))-1
31 :
32 :  #define DI()            asm("FCLR I")   // 割り込み禁止
33 :  #define EI()            asm("FSET I")   // 割り込み許可
```

名称	説明
TIMER_CYCLE	TYMER_CYCLE は、タイマ RB の割り込みを発生させる間隔を設定します。 今回は 1［ms］に設定しますので、 （1×10^{-3}）÷（1÷（$20\times10^{6}\div128$））－1＝155 となります。
PWM_CYCLE	PWM_CYCLE は、左右モーターに加えるタイマ RD の PWM 周期を設定します。 今回は 16［ms］に設定しますので、 （16×10^{-3}）÷（1÷（$20\times10^{6}\div8$））－1＝39999 となります。
Def_500Hz	Def_500Hz は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 500［Hz］に設定しますので、 （1÷500）÷（1÷（$20\times10^{6}\div8$））－1＝4999 となります。
Def_1000Hz	Def_1000Hz は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 1000［Hz］に設定しますので、 （1÷1000）÷（1÷（$20\times10^{6}\div8$））－1＝2499 となります。
Def_C3	Def_C3 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 131［Hz］に設定しますので、 （1÷131）÷（1÷（$20\times10^{6}\div8$））－1＝19083 となります。
Def_D3	Def_D3 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 147［Hz］に設定しますので、 （1÷147）÷（1÷（$20\times10^{6}\div8$））－1＝17006 となります。
Def_E3	Def_E3 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 165［Hz］に設定しますので、 （1÷165）÷（1÷（$20\times10^{6}\div8$））－1＝15151 となります。
Def_F3	Def_F3 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 175［Hz］に設定しますので、 （1÷175）÷（1÷（$20\times10^{6}\div8$））－1＝14285 となります。

Def_G3	Def_G3 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 196［Hz］に設定しますので、 （1÷196）÷（1÷（20×10^6÷8））－1＝12754 となります。
Def_A3	Def_A3 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 220［Hz］に設定しますので、 （1÷220）÷（1÷（20×10^6÷8））－1＝11362 となります。
Def_B3	Def_B3 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 247［Hz］に設定しますので、 （1÷247）÷（1÷（20×10^6÷8））－1＝10120 となります。
Def_C4	Def_C4 は、圧電サウンダに加えるタイマ RC の PWM 周期を設定します。 今回は 262［Hz］に設定しますので、 （1÷262）÷（1÷（20×10^6÷8））－1＝9541 となります。
DI()	DI()は、割り込み禁止のインラインアセンブルの定義です。
EI()	DI()は、割り込み許可のインラインアセンブルの定義です。

5.5 関数プロトタイプ

プログラム

```
38 :  void init( void );
39 :  unsigned char sensor( void );
40 :  void motor( int data1, int data2 );
41 :  void timer( unsigned long timer_set );
42 :  void beep( int data1 );
43 :  unsigned char dipsw( void );
44 :  unsigned char pushsw( void );
```

関数プロトタイプとは、関数の引数の型と個数をチェックするために、関数を使用する前に宣言する部分のことです。関数プロトタイプは、関数に「;」を付加したものです。

5.6 グローバル変数

プログラム

```
 49 :  unsigned long   cnt0 = 0;              // timer 関数用
 50 :  unsigned long   cnt1 = 0;              // main 内で使用
 51 :  int             pattern = 0;           // パターン番号
```

グローバル変数とは、関数の外で定義されている、どの関数からも参照できる変数のことです。

ローカル変数とは、関数の中で定義されている、関数の中でのみ参照できる変数のことです。

以下に例を示します。

参考例

```
void a( void );                         // プロトタイプ宣言

int timer;                              // グローバル変数

void main( void )
{
          int i;

          timer = 0;
          i = 10;
          printf( “%d¥n”,timer );       // ←0 を表示
          a();
          printf( “%d¥n”,timer );       // ←timer はグローバル変数なので、
                                        //   a 関数内でセットした 20 を表示
          printf( “%d¥n”,i );           // ←a 関数でも変数 i を使っているがローカル
                                        //   変数なので、a 関数内の i 変数は無関係
                                        //   この関数でセットした 10 が表示される
}

void a( void )
{
          int i;
          i = 20;
          timer = i;
}
```

mini_mcr.c では、3 つのグローバル変数を宣言しています。

名称	型	説明
cnt0	unsigned long	timer 関数で時間を計る（1[ms]単位）ときに使用します。
cnt1	unsigned long	main 関数などで時間を計る（1[ms]単位）ときに使用します。
pattern	int	パターン番号です。

5.7 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初めに行います。

5.8 R8C/35A の内蔵周辺機能の初期化：init 関数

R8C/35A の内蔵周辺機能の初期化を行います。
周辺機能の初期化を行う際には、割り込みを禁止にし、モジュールストップは解除しておきます。

5.8.1 クロック発生回路の XIN クロック設定

初めに、クロック発生回路の初期化を行います。

プログラム

```
457 :           prc0 = 1;
458 :
459 :           cm13 = 1;
460 :           cm05 = 0;
461 :           while(i <= 50) i++;
462 :           ocd2 = 0;
463 :
464 :           prc0 = 0;
```

レジスタ	ビット	シンボル	説明
PRCR	0	PRC0	CM0、CM1 レジスタへの書き込みを許可するため、始めに“1”にします。最後は“0”に戻します。
CM1	3	CM13	端子を XIN-XOUT 端子として使用するため、“1”にします。
CM0	5	CM05	XIN クロックを発振させるため、“1”にします。
OCD	2	OCD2	システムクロックを XIN クロックにするため、“0”にします。

5.8.2 I/O ポートの入出力設定

プログラム

```
467 :          prc2 = 1;                // pd0 レジスタへの書き込み許可
468 :          pd0 = 0xe0;              // P0_0～P0_3:センサー
469 :                                   // P0_4:マイクロスイッチ
470 :                                   // P0_5～P0_7:LED
471 :          prc2 = 0;                // pd0 レジスタへの書き込み禁止
472 :          pd1 = 0xdf;              // P1_0～P1_3:LED
473 :                                   // P1_4:TXD0
474 :                                   // P1_5:RXD0
475 :          pd2 = 0xfe;              // P2_0:スイッチ
476 :                                   // P2_1:AIN1
477 :                                   // P2_2:PWMA
478 :                                   // P2_3:BIN1
479 :                                   // P2_4:PWMB
480 :                                   // P2_5:SERVO
481 :                                   // P2_6:AIN2
482 :                                   // P2_7:BIN2
483 :          pd3 = 0xfb;              // P3_2:赤外線受信
484 :                                   // P3_4:ブザー
485 :          pd4 = 0x80;              // P4_2:VREF
486 :                                   // P4_3～P4_5:DIPSW
487 :                                   // P4_6:XIN
488 :                                   // P4_7:XOUT
489 :          pd5 = 0x40;              // P5_7:DIPSW
490 :          pd6 = 0xff;              //
```

入出力の決め方

出力	出力端子は出力に設定します。
入力	入力端子は入力に設定します。
未接続	未接続端子は出力に設定します。
-	端子のないビットは入力に設定します。

ポート	ビット	接続先	入出力	設定値 PDi
0	7	LEDC 出力	出力	0xe0
	6	LEDB 出力	出力	
	5	LEDA 出力	出力	
	4	マイクロスイッチ入力	入力	
	3	赤外線フォトインタラプタ 3 入力	入力	
	2	赤外線フォトインタラプタ 2 入力	入力	
	1	赤外線フォトインタラプタ 1 入力	入力	
	0	赤外線フォトインタラプタ 0 入力	入力	
1	7	未接続	出力	0xdf
	6	未接続	出力	
	5	RXD0 入力	入力	
	4	TXD0 出力	出力	
	3	LED3 出力	出力	
	2	LED2 出力	出力	
	1	LED1 出力	出力	
	0	LED0 出力	出力	

ポート	ビット	接続先	入出力	設定値 PDi
2	7	モーター右 2 出力	出力	0xfe
	6	モーター左 2 出力	出力	
	5	サーボ出力	出力	
	4	モーター右 PWM 出力	出力	
	3	モーター右 1 出力	出力	
	2	モーター左 PWM 出力	出力	
	1	モーター左 1 出力	出力	
	0	タクトスイッチ入力	入力	
3	7	未接続	出力	0xfb
	6	未接続	出力	
	5	未接続	出力	
	4	圧電サウンダ	出力	
	3	未接続	出力	
	2	赤外線リモコン受光モジュール	入力	
	1	未接続	出力	
	0	未接続	出力	
4	7	XOUT 出力	出力	0x80
	6	XIN 入力	入力	
	5	DIP スイッチ入力	入力	
	4	DIP スイッチ入力	入力	
	3	DIP スイッチ入力	入力	
	2	VREF 入力	入力	
	1	-	入力	
	0	-	入力	
5	7	DIP スイッチ入力	入力	0x40
	6	未接続	出力	
	5	-	入力	
	4	-	入力	
	3	-	入力	
	2	-	入力	
	1	-	入力	
	0	-	入力	
6	7	未接続	出力	0xff
	6	未接続	出力	
	5	未接続	出力	
	4	未接続	出力	
	3	未接続	出力	
	2	未接続	出力	
	1	未接続	出力	
	0	未接続	出力	

PD0 レジスタを設定するには、PRCP レジスタの PRC2 ビットを“1”にする必要があります。

5.8.3 タイマ RB の 1［ms］割り込み設定

プログラム

```
499 :       trbmr = 0x00;                   // カウントソースは f1
500 :       trbpre = 128 - 1;               // プリスケーラ
501 :       trbpr = TIMER_CYCLE;            // プライマリカウンタ
502 :       trbic = 0x01;                   // タイマ RB の割り込みレベル設定
503 :       trbcr = 0x01;                   // カウントを開始
```

レジスタ	ビット	シンボル	説明	設定値
TRBMR	7	TCKCUT	カウントソースを供給するため、“0”にします。	0x00
	6	-	何も配置されていないので、“0”にします。	
	5	TCK1	カウントソースを f1 にするため、“00”にします。	
	4	TCK0		
	3	TWRC	リロードレジスタとカウンタへの書き込みを選択するため、“0”にします。	
	2	-	何も配置されていないので、“0”にします。	
	1	TMOD1	タイマーモードにするため、“0”にします。	
	0	TMOD0		
TRBPRE	7-0	-	内部カウントソースをカウントします。 この値よりカウントが行われ、アンダーフローすると、TRBPR がカウントされます。	128-1
TRBPR	7-0	-	《TIMER_CYCLE》 TRBPRE レジスタのアンダーフローをカウントします。 この値よりカウントが行われ、アンダーフローすると、割り込みが発生します。 値の計算式は、 t＝設定時間、f1＝クリスタル周波数、pre＝分周比（TRBPRE＋1） $t \div \frac{1}{f1 \div pre} - 1$ 1［ms］単位で割り込みを行いますので、 $1\times10^{-3} \div \frac{1}{20\times10^{6} \div 128} - 1 = 155$ となります。	155
TRBIC	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をクリアするため、“0”にします。	
	2	ILVL2	割り込みレベルを、“1”にします。	
	1	ILVL1		
	0	ILVL0		
TRBCR	7	-	何も配置されていないので、“0”にします。	0x01
	6	-		
	5	-		
	4	-		
	3	-		
	2	TSTOP	カウントを強制停止させませんので、“0”にします。	
	1	TCSTF	読み込み専用ですが、“0”にしておきます。	
	0	TSTART	カウントを開始するため、“1”にします。	

5.8.4 タイマ RC の PWM モード

プログラム

```
506 :        trccr1 = 0xb0;          // カウントソースは f8
507 :        trcgra = 0;             // 圧電サウンダの周期
508 :        trcgrc = 0;             // 圧電サウンダのデューティ比
509 :        trccr2 = 0x02;          // TRCIOC 端子はアクティブレベル H
510 :        trcoer = 0x0b;          // TRCIOC 端子の出力許可
511 :        trcpsr1 = 0x02;         // TRCIOC 端子を P3_4 に割り当て
512 :        trcmr = 0x8a;           // カウントを開始
```

レジスタ	ビット	シンボル	説明	設定値
TRCCR1	7	CCLR	TRCGRA レジスタのコンペア一致で TRC レジスタをクリアさせるため、“1”にします。	0xb0
	6	TCK2	カウントソースを f8 にするため、“011”にします。	
	5	TCK1		
	4	TCK0		
	3	TOD	使用しません。“0”にしておきます。	
	2	TOC	TRCIOC 端子の初期出力をアクティブではないレベルにするため、“0”にします。	
	1	TOB	使用しません。“0”にしておきます。	
	0	TOA	PWM モードでは無効なので、“0”にします。	
TRCGRA	15-0	-	音の周波数を決めます。 最初は音を出さないため、“0”にします。 値の計算式は、 fs＝音の周波数、f8＝クリスタル周波数÷8 $\frac{1}{fs} \div \frac{1}{f8} - 1$ 1［KHz］の音を出す場合は、 $\frac{1}{1\times10^{3}} \div \frac{1}{20\times10^{6} \div 8} - 1 = 2499$ となります。	0
TRCGRC	15-0	-	音のデューティ比は TRCGRA レジスタの半分の値を入れます。 最初は音を出さないため、“0”にします。	0
TRCCR2	7	TCEG1	PWM モードでは無効なので、“00”にします。	0x02
	6	TCEG0		
	5	CSEL	TRCGRA レジスタとのコンペア一致後もカウントを継続させるため、“0”にします。	
	4	-	何も配置されていないので、“0”にします。	
	3	-	何も配置されていないので、“0”にします。	
	2	POLD	使用しません。“0”にしておきます。	
	1	POLC	TRCIOC 端子をアクティブレベル H にするため、“1”にします。	
	0	POLB	使用しません。“0”にしておきます。	
TRCOER	7	PTO	パルス出力強制遮断入力を無効にしますので、“0”にします。	0x0b
	6	-	何も配置されていないので、“0”にします。	
	5	-		
	4	-		
	3	ED	TRCIOD 端子を出力禁止にするため、“1”にします。	
	2	EC	TRCIOC 端子を出力許可にするため、“0”にします。	
	1	EB	TRCIOB 端子を出力禁止にするため、“1”にします。	
	0	EA	TRCIOA 端子を出力禁止にするため、“1”にします。	

レジスタ	ビット	シンボル	説明	設定値
TRCPSR1	7	-	何も配置されていないので、“0”にします。	0x02
	6	TRCIODSEL2	TRCIOD 端子は使用しないので、“000”にします。	
	5	TRCIODSEL1		
	4	TRCIODSEL0		
	3	-	何も配置されていないので、“0”にします。	
	2	TRCIOCSEL2	TRCIOC 端子を P3_4 に割り当てるので、“010”にします。	
	1	TRCIOCSEL1		
	0	TRCIOCSEL0		
TRCMR	7	TSTART	カウントを開始するため、“1”にします。	0x8a
	6	-	何も配置されていないので、“0”にします。	
	5	BFD	TRCGRD レジスタをジェネラルレジスタにするために、“0”にします。	
	4	BFC	TRCGRC レジスタをジェネラルレジスタにするために、“0”にします。	
	3	PWM2	PWM モードにするために、“1”にします。	
	2	PWMD	使用しません。“0”にしておきます。	
	1	PWMC	TRCIOC 端子を PWM モードにするために、“1”にします。	
	0	PWMB	使用しません。“0”にしておきます。	

5.8.5 タイマ RD のリセット同期 PWM モード

プログラム

```
515 :          trdpsr0 = 0x08;                  // TRDIOB0 端子を P2_2 に割り当て
516 :          trdpsr1 = 0x05;                  // TRDIOB1 端子を P2_5 に割り当て
517 :                                           // TRDIOA1 端子を P2_4 に割り当て
518 :          trdmr = 0xf0;                    // レジスタをバッファ動作にする
519 :          trdfcr = 0x01;                   // リセット同期 PWM モードに設定
520 :          trdoer1 = 0xcd;                  // TRDIOB1 の出力許可
521 :                                           // TRDIOA1 の出力許可
522 :                                           // TRDIOB0 端子の出力許可
523 :          trdcr0 = 0x23;                   // カウントソースは f8
524 :          trdgra0 = trdgrc0 = PWM_CYCLE;   // 周期
525 :          trdgrb0 = trdgrd0 = 0;           // TRDIOB0 端子（左モータ）
526 :          trdgra1 = trdgrc1 = 0;           // TRDIOA1 端子（右モータ）
527 :          trdgrb1 = trdgrd1 = 0;           // TRDIOB1 端子（サーボ）
528 :          trdstr = 0x0d;                   // カウントを開始
```

レジスタ	ビット	シンボル	説明	設定値
TRDPSR0	7	-	何も配置されていないので、“0”にします。	0x08
	6	TRDIOD0SEL0	TRDIOD0 端子は使用しないので、“0”にします。	
	5	TRDIOC0SEL1	TRDIOC0 端子は使用しないので、“00”にします。	
	4	TRDIOC0SEL0		
	3	TRDIOB0SEL1	TRDIOB0 端子を P2_2 に割り当てるので、“10”にします。	
	2	TRDIOB0SEL0		
	1	-	何も配置されていないので、“0”にします。	
	0	TRDIOA0SEL0	TRDIOA0 端子は使用しないので、“0”にします。	
TRDPSR1	7	-	予約ビットです。“0”にします。	0x05
	6	TRDIOD1SEL0	TRDIOD1 端子は使用しないので、“0”にします。	
	5	-	予約ビットです。“0”にします。	
	4	TRDIOC1SEL0	TRDIOC1 端子は使用しないので、“0”にします。	
	3	-	何も配置されていないので、“0”にします。	
	2	TRDIOB1SEL0	TRDIOB1 端子を P2_5 に割り当てるので、“1”にします。	
	1	-	何も配置されていないので、“0”にします。	
	0	TRDIOA1SEL0	TRDIOA1 端子を P2_4 に割り当てるので、“1”にします。	
TRDMR	7	BFD1	TRDGRD1 を TRDGRB1 のバッファレジスタにするため、“1”にします。	0xf0
	6	BFC1	TRDGRC1 を TRDGRA1 のバッファレジスタにするため、“1”にします。	
	5	BFD0	TRDGRD0 を TRDGRB0 のバッファレジスタにするため、“1”にします。	
	4	BFC0	TRDGRC0 を TRDGRA0 のバッファレジスタにするため、“1”にします。	
	3	-	何も配置されていないので、“0”にします。	
	2	-		
	1	-		
	0	SYNC	リセット同期 PWM モードでは、“0”にします。	
TRDFCR	7	PWM3	リセット同期 PWM モードでは無効なので、“0”にします。	0x01
	6	STCLK	外部クロック入力を無効にするので、“0”にします。	
	5	ADEG	リセット同期 PWM モードでは無効なので、“0”にします。	
	4	ADTRG		
	3	OLS1	初期出力 H、アクティブレベル L にしますので、“00”にします。	
	2	OLS0		
	1	CMD1	リセット同期 PWM モードでは、“01”にします。	
	0	CMD0		

レジスタ	ビット	シンボル	説明	設定値
TRDOER1	7	ED1	TRDIOD1 端子を出力禁止にするため、“1”にします。	0xcd
	6	EC1	TRDIOC1 端子を出力禁止にするため、“1”にします。	
	5	EB1	TRDIOB1 端子を出力許可にするため、“0”にします。	
	4	EA1	TRDIOA1 端子を出力許可にするため、“0”にします。	
	3	ED0	TRDIOD0 端子を出力禁止にするため、“1”にします。	
	2	EC0	TRDIOC0 端子を出力禁止にするため、“1”にします。	
	1	EB0	TRDIOB0 端子を出力許可にするため、“0”にします。	
	0	EA0	TRDIOA0 端子を出力禁止にするため、“1”にします。	
TRDCR0	7	CCLR2	リセット同期 PWM モードでは、“001”にします。	0x23
	6	CCLR1		
	5	CCLR0		
	4	CKEG1	使用しません。“00”にしておきます。	
	3	CKEG0		
	2	TCK2	カウントソースを f8 にするため、“011”にします。	
	1	TCK1		
	0	TCK0		
TRDGRA0 TRDGRC0	15-0	-	《PWM_CYCLE》 PWM 周期を設定します。 値の計算式は、 t＝設定時間、f8＝クリスタル周波数÷8 $t \div \frac{1}{f8} - 1$ 周期を 16［ms］にしますので、 $16\times10^{-3} \div \frac{1}{20\times10^{6} \div 8} - 1 = 39999$ となります。	39999
TRDGRB0 TRDGRD0	15-0	-	最初は左モーターを動かさないため、“0”にします。 バッファ動作のため TRDGRD0 レジスタにも同じ値を入れます。	0
TRDGRA1 TRDGRC1	15-0	-	最初は右モーターを動かさないため、“0”にします。 バッファ動作のため TRDGRC1 レジスタにも同じ値を入れます。	0
TRDGRB1 TRDGRD1	15-0	-	最初はサーボを動かさないため、“0”にします。 バッファ動作のため TRDGRD1 レジスタにも同じ値を入れます。	0
TRDSTR	7	-	何も配置されていないので、“0”にします。	0x0d
	6	-		
	5	-		
	4	-		
	3	CSEL1	TRDGRA1 レジスタとのコンペア一致後もカウントを継続させますので、“1”にします。	
	2	CSEL0	TRDGRA0 レジスタとのコンペア一致後もカウントを継続させますので、“1”にします。	
	1	TSTART1	使用しません。“0”にしておきます。	
	0	TSTART0	カウントを開始するため、“1”にします。	

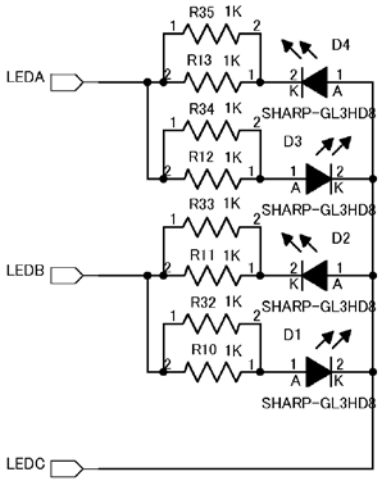
5.9 割り込みプログラム：intTRBIC 関数

intTRBIC 関数は、1［ms］ごとに割り込みで実行されます。

プログラム

```
537 :  #pragma interrupt intTRBIC (vect=24)
538 :  void intTRBIC( void )
539 :  {
540 :          p0_7 = ~p0_7;
541 :
542 :          if( p0_7 == 0 ){
543 :                  //p0_1、p0_3 のモニタが可能
544 :                  p0_5 = ~p0_1;
545 :                  p0_6 = ~p0_3;
546 :          }else{
547 :                  //p0_0、p0_2 のモニタが可能
548 :                  p0_5 = p0_0;
549 :                  p0_6 = p0_2;
550 :          }
551 :
552 :          cnt0++;
553 :          cnt1++;
554 :  }
```

回路図



```
537 :  #pragma interrupt intTRBIC (vect=24)
```

#pragma interrupt は、割り込み関数の名称とベクターアドレスを定義します。

```
540 :          p0_7 = ~p0_7;
```

P0_7 端子の出力信号を反転させています。

```
542 :          if( p0_7 == 0 ){
543 :                  //p0_1、p0_3 のモニタが可能
544 :                  p0_5 = ~p0_1;
545 :                  p0_6 = ~p0_3;
546 :          }else{
547 :                  //p0_0、p0_2 のモニタが可能
548 :                  p0_5 = p0_0;
549 :                  p0_6 = p0_2;
550 :          }
```

P0_7 端子の状態を読み込み、センサーの状態をモニターする LED の点灯制御を切り替えています。

端子	レベル	端子	レベル	説明
P0_7	H	P0_6	L	D2 の LED が点灯します。
		P0_5	L	D4 の LED が点灯します。
	L	P0_6	H	D1 の LED が点灯します。
		P0_5	H	D3 の LED が点灯します。

```
552 :          cnt0++;
553 :          cnt1++;
```

cnt0 変数を+1 しています。この変数の値をチェックすることにより、1[ms]単位の時間の計測が行えます。cnt0 変数と同様に、cnt1 変数を+1 しています。

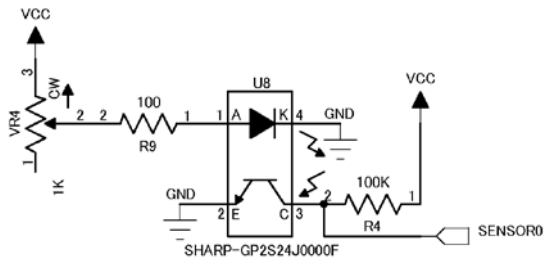
5.10 センサー状態検出：sensor 関数

sensor 関数は、センサー（赤外線フォトインタラプタ）の状態を検出します。

プログラム

```
561 :  unsigned char sensor( void )
562 :  {
563 :          volatile unsigned char  data1;
564 :
565 :          data1 = ~p0;                    // ラインの色は白
566 :          data1 = data1 & 0x0f;
567 :
568 :          return( data1 );
569 :  }
```

回路図




```
565 :          data1 = ~p0;
```

P0 レジスタを読み込み、反転します。センサーはポート 0 の端子につながっていますので、P0 レジスタを読み込むことにより、状態を検出できます。白いラインがある場合に、センサーの赤外線は反射され、ポート 0 の端子は L になります。ラインがある場合に“1”にしたいので、反転をします。黒いラインを使用する場合は反転の必要はありません。

```
566 :          data1 = data1 & 0x0f;
```

マスクをかけます。P0 レジスタを読み込む場合、8 ビット単位で読み込まれます。センサーはポート 0 の 0～3 の端子にしかつながっていませんので、P0 レジスタの 4～7 ビットには必要のない値が入っています。そこで、0x0f と AND をとることにより、4～7 ビットを“0”にします。

```
568 :          return( data1 );
```

関数の呼び出し元に値を返します。

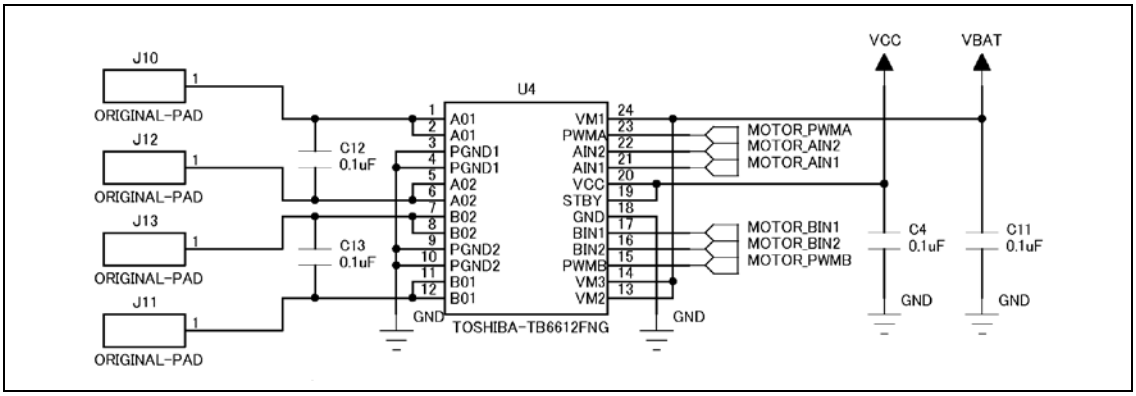
5.11 モーター速度制御：motor 関数

motor 関数は、引数で指定したデューティ比で左右のモーターを動かします。

プログラム

```
577 :  void motor( int data1, int data2 )
578 :  {
579 :          volatile int    motor_r;
580 :          volatile int    motor_l;
581 :          volatile int    sw_data;
582 :
583 :          sw_data = dipsw() + 5;
584 :          motor_l = (long)data1 * sw_data / 20;
585 :          motor_r = (long)data2 * sw_data / 20;
586 :
587 :          if( motor_l >= 0 ) {
588 :                  p2_1 = 0;
589 :                  p2_6 = 1;
590 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
591 :          } else {
592 :                  p2_1 = 1;
593 :                  p2_6 = 0;
594 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
595 :          }
596 :
597 :          if( motor_r >= 0 ) {
598 :                  p2_3 = 0;
599 :                  p2_7 = 1;
600 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
601 :          } else {
602 :                  p2_3 = 1;
603 :                  p2_7 = 0;
604 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
605 :          }
606 :  }
```

回路図




```
583 :          sw_data = dipsw() + 5;
```

dipsw 関数は、DIP スイッチの値が返ってきます。返ってくる値は 0～15 です。返ってきた値に +5 していますので、sw_data 変数には 5～20 の値が入ることになります。

```
584 :           motor_l = (long)data1 * sw_data / 20;
585 :           motor_r = (long)data2 * sw_data / 20;
```

引数で指定したデューティ比に、DIP スイッチの値で設定した比率を掛け合わせます。

$$\text{引数で指定したデューティ比} \quad \times \frac{\text{sw\_data}}{20}$$

DIP スイッチ（ON：0、OFF：1）				10 進数	計算	モータースピードの割合
P5_7（3）	P4_5（2）	P4_4（1）	P4_3（0）			
0	0	0	0	0	5/20	25%
0	0	0	1	1	6/20	30%
0	0	1	0	2	7/20	35%
0	0	1	1	3	8/20	40%
0	1	0	0	4	9/20	45%
0	1	0	1	5	10/20	50%
0	1	1	0	6	11/20	55%
0	1	1	1	7	12/20	60%
1	0	0	0	8	13/20	65%
1	0	0	1	9	14/20	70%
1	0	1	0	10	15/20	75%
1	0	1	1	11	16/20	80%
1	1	0	0	12	17/20	85%
1	1	0	1	13	18/20	90%
1	1	1	0	14	19/20	95%
1	1	1	1	15	20/20	100%

```
587 :          if( motor_l >= 0 ) {
588 :                  p2_1 = 0;
589 :                  p2_6 = 1;
590 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * motor_l / 100;
591 :          } else {
592 :                  p2_1 = 1;
593 :                  p2_6 = 0;
594 :                  trdgrd0 = (long)( PWM_CYCLE - 1 ) * ( -motor_l ) / 100;
595 :          }
596 :
597 :          if( motor_r >= 0 ) {
598 :                  p2_3 = 0;
599 :                  p2_7 = 1;
600 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * motor_r / 100;
601 :          } else {
602 :                  p2_3 = 1;
603 :                  p2_7 = 0;
604 :                  trdgrc1 = (long)( PWM_CYCLE - 1 ) * ( -motor_r ) / 100;
605 :          }
```

回転方向とデューティ比を設定しています。

端子			説明
P2_1	P2_6	TRDIOB0	
H	H	H／L	左モーターショートブレーキ
L	H	H	左モーター正転
		L	左モーターショートブレーキ
H	L	H	左モーター逆転
		L	左モーターショートブレーキ
L	L	H	左モーター惰性

端子			説明
P2_3	P2_7	TRDIOA1	
H	H	H／L	右モーターショートブレーキ
L	H	H	右モーター正転
		L	右モーターショートブレーキ
H	L	H	右モーター逆転
		L	右モーターショートブレーキ
L	L	H	右モーター惰性

TRDGRD0、TRDGRC1 レジスタに設定した値によって、デューティ比が決まります。

$$(\ \mathrm{PWM\_CYCLE} \ - \ 1 \ ) \ \times \frac{\mathrm{motor\_l}}{100}$$

PWM_CYCLE から-1 しているのは、motor_l が 100 になったとき、PWM_CYCLE と同じ値にならないようにするためです。同じ値になると、TRDGRA0、TRDGRD0 レジスタのコンペア一致が同時に起こり、TRDGRD0 レジスタのコンペア一致が優先され、初期出力の H にならずアクティブレベルの L になったままになってしまうためです。

5.12 時間稼ぎ：timer 関数

timer 関数は、cnt0 変数が、引数で指定した値より大きくなるまで、時間稼ぎをします。

プログラム

```
613 :  void timer( unsigned long data1 )
614 :  {
615 :          cnt0 = 0;
616 :          while( cnt0 < data1 );
617 :  }
```

```
615 :          cnt0 = 0;
```

初めに cnt0 変数をクリアしておきます。

```
616 :          while( cnt0 < data1 );
```

cnt0 変数が、割り込みで 1［ms］ごとに+1 されますので、指定した時間がたつと while 文から抜け出します。

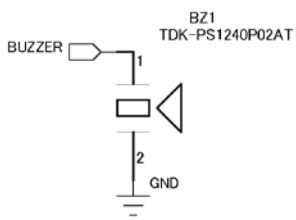
5.13 音を鳴らす：beep 関数

beep 関数は、引数で指定した値の周期で 50%の PWM 信号を出力し、音を出します。

プログラム

```
624 :  void beep( int data1 )
625 :  {
626 :          trcgra = data1;                 // 周期の設定
627 :          trcgrc = data1 / 2;             // デューティ 50%のため周期の半分の値
628 :  }
```

回路図




```
626 :          trcgra = data1;                 // 周期の設定
```

周期の設定をします。

```
627 :          trcgrc = data1 / 2;             // デューティ 50%のため周期の半分の値
```

デューティ比は 50%にするため、周期の半分の値を入れます。

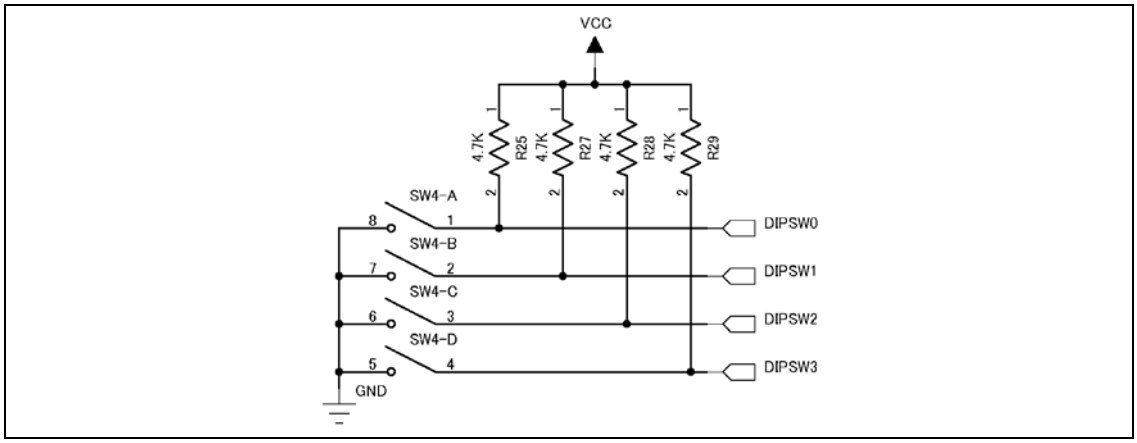
5.14 DIP スイッチ状態検出：dipsw 関数

dipsw 関数は、DIP スイッチが ON のときに“0”、OFF のときに“1”の値を返します。

プログラム

```
635 :  unsigned char dipsw( void )
636 :  {
637 :          volatile unsigned char  data1;
638 :
639 :          data1 = ( ( p5 >> 4 ) & 0x08 ) | ( ( p4 >> 3 ) & 0x07 );
640 :
641 :          return( data1 );
642 :  }
```

回路図




```
639 :          data1 = ( ( p5 >> 4 ) & 0x08 ) | ( ( p4 >> 3 ) & 0x07 );
```

DIP スイッチは P5_7、P4_5、P4_4、P4_3 の端子につながっています。これらのデータを合わせて、ひとつにします。

P5 レジスタの読み込み

P5_7	X	X	X	X	X	X	X

P5 レジスタを 4 ビット右にシフト

0	0	0	0	P5_7	X	X	X

0x08 と AND

0	0	0	0	1	0	0	0

①

0	0	0	0	P5_7	0	0	0

P4 レジスタの読み込み

X	X	P4_5	P4_4	P4_3	X	X	X

P4 レジスタを 3 ビット右にシフト

0	0	0	X	X	P4_5	P4_4	P4_3

0x07 と AND

0	0	0	0	0	1	1	1

②

0	0	0	0	0	P4_5	P4_4	P4_3

①＋②

0	0	0	0	P5_7	P4_5	P4_4	P4_3

```
641 :           return( data1 );
```

関数の呼び出し元に値を返します。

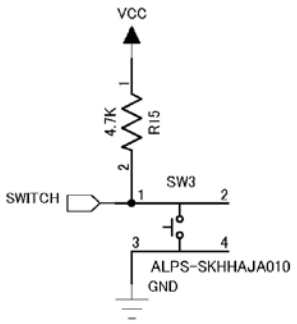
5.15 プッシュスイッチ状態検出：pushsw 関数

pushsw 関数は、プッシュスイッチ（タクトスイッチ）が OFF のときに“0”、ON のときに“1”の値を返します。

プログラム

```
649 :  unsigned char pushsw( void )
650 :  {
651 :          unsigned char data1;
652 :
653 :          data1 = ~p2;
654 :          data1 &= 0x01;
655 :
656 :          return( data1 );
657 :  }
```

回路図




```
653 :          data1 = ~p2;
```

P2 レジスタを読み込み、反転します。プッシュスイッチは、ポート 2 の端子につながっていますので、P2 レジスタを読み込むことにより、状態を検出できます。プッシュスイッチを押した場合、GND とショート状態になり、ポート 2 の端子は L になります。プッシュスイッチを押した場合に“1”にしたいので、反転をします。

```
654 :          data1 &= 0x01;
```

マスクをかけます。P2 レジスタを読み込む場合、8 ビット単位で読み込まれます。プッシュスイッチは、ポート 2 の 0 の端子にしかつながっていませんので、P2 レジスタの 1～7 ビットには必要のない値が入っています。そこで、0x01 と AND をとることにより、1～7 ビットを“0”にします。

```
656 :          return( data1 );
```

関数の呼び出し元に値を返します。

5.16 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。

5.16.1 起動時実行部分

プログラム

```
 56 :  void main(void)
 57 :  {
 58 :          // 初期化
 59 :          init();
 60 :
 61 :          // 起動音
 62 :          beep(Def_500Hz);
 63 :          timer(100);
 64 :          beep(Def_1000Hz);
 65 :          timer(100);
 66 :          beep(0);
 68～442 :  「(1) プログラム」を参照。
444 :  }
```

```
 59 :          init();
```

init 関数を実行し、R8C/35A の内蔵周辺機能の初期化を行います。

```
 62 :          beep(Def_500Hz);
 63 :          timer(100);
 64 :          beep(Def_1000Hz);
 65 :          timer(100);
 66 :          beep(0);
```

起動音を出します。500［Hz］の音を 0.1 秒間、1000［Hz］の音を 0.1 秒間出した後、音を止めています。

(1) プログラム

```
 68 :          while(1){
 69 :          switch( pattern ){
 86～ 94 :  「5.16.3  パターン 0：スイッチ入力待ち」を参照。
 96～104 :  「5.16.4  パターン 1：1 秒後にスタート」を参照。
106～172 :  「5.16.5  パターン 11：通常トレース」を参照。
174～241 :  「5.16.6  パターン 21：クロスライン検出後のトレース、クランク検出」を参照。
243～249 :  「5.16.7  パターン 22：クランクの曲げ動作継続処理」を参照。
251～318 :  「5.16.8  パターン 31：左ハーフライン検出後のトレース、左レーンチェンジ検出」を参照。
320～328 :  「5.16.9  パターン 32：左レーンチェンジ曲げ動作継続処理」を参照。
330～343 :  「5.16.9  パターン 33：左レーンチェンジ終了検出」を参照。
345～412 :  「5.16.10  パターン 41：右ハーフライン検出後のトレース、右レーンチェンジ検出」を参照。
414～422 :  「5.16.11  パターン 42：右レーンチェンジ曲げ動作継続処理」を参照。
424～437 :  「5.16.12  パターン 43：右レーンチェンジ終了検出」を参照。
439 :          default:
440 :                  break;
441 :
442 :          }
443 :          }
```

while 文は、() 内の式が“真“なら {} 内の文を繰り返し実行し、「偽」なら {} の次の文から実行する制御文です。while 文の () 内の式が“1”の場合、常に「真」となるので、() 内の文を永久に繰り返し実行します。switch 文では、pattern 変数の数値によって、case 文が分岐します。

5.16.2 パターン

パターン	状態	終了条件
0	スイッチ入力待ち	・スイッチを押した場合、パターン 1 へ
1	1 秒後にスタート	・1000［ms］たった場合、パターン 11 へ
11	通常トレース	・クロスラインを検出した場合、パターン 21 へ ・左ハーフラインを検出した場合、パターン 31 へ ・右ハーフラインを検出した場合、パターン 41 へ
21	クロスライン検出後のトレース、 クランク検出	・クランクを検出した場合、パターン 22 へ
22	クランクの曲げ動作継続処理	・1000［ms］たった場合、パターン 11 へ
31	左ハーフライン検出後のトレース、 左レーンチェンジ検出	・左レーンチェンジを検出した場合、パターン 32 へ ・クロスラインを検出した場合、パターン 21 へ
32	左レーンチェンジ曲げ動作継続処理	・700［ms］たった場合、パターン 33 へ
33	左レーンチェンジ終了検出	・右端のセンサーのみ反応した場合、パターン 11 へ
41	右ハーフライン検出後のトレース、 右レーンチェンジ検出	・右レーンチェンジを検出した場合、パターン 42 へ ・クロスラインを検出した場合、パターン 21 へ
42	右レーンチェンジ曲げ動作継続処理	・700［ms］たった場合、パターン 43 へ
43	右レーンチェンジ終了検出	・左端のセンサーのみ反応した場合、パターン 11 へ

5.16.3 パターン 0：スイッチ入力待ち

プログラム

```
 86 :          case 0:
 87 :                  // スイッチ入力待ち
 88 :                  if( pushsw() == 1 ){
 89 :                          beep(Def_1000Hz);
 90 :                          cnt1 = 0;
 91 :                          pattern = 1;
 92 :                  }
 93 :
 94 :                  break;
```

if 文では、pushsw 関数の戻り値が“1”の（スイッチが押された）場合、{} 内の文を実行します。{} 内では、1［KHz］の音を出し、cnt1 変数をクリアして、パターン 1 に行きます。

5.16.4 パターン 1：1 秒後にスタート

プログラム

```
 96 :          case 1:
 97 :                  // 1 秒後にスタート
 98 :                  if( cnt1 >= 1000 ){
 99 :                          beep(0);
100 :                          cnt1 = 0;
101 :                          pattern = 11;
102 :                  }
103 :
104 :                  break;
```

if 文では、cnt1 変数が 1000 以上の（1000［ms］経過した）場合、{} 内の文を実行します。{} 内では、音を止め、cnt1 変数をクリアして、パターン 11 に行きます。

5.16.5 パターン 11：通常トレース

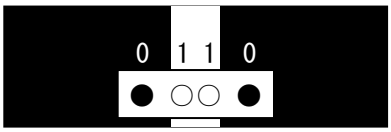
プログラム

```
106 :          case 11:
107 :                  // 通常トレース
108 :                  beep(0);
109 :
110 :                  switch( ( sensor() & 0x0f ) ){
111～114 :  「(1) 中央を走行しているとき」を参照。
116～119 :  「(2) 少し右側を走行しているとき」を参照。
121～124 :  「(3) 中くらい右側を走行しているとき」を参照。
126～129 :  「(4) 大きく右側を走行しているとき」を参照。
131～134 :  「(5) 少し左側を走行しているとき」を参照。
136～139 :  「(6) 中くらい左側を走行しているとき」を参照。
141～144 :  「(7) 大きく左側を走行しているとき」を参照。
146～151 :  「(8) クロスラインを検出しているとき」を参照。
153～158 :  「(9) 左ハーフラインを検出しているとき」を参照。
160～165 :  「(10) 右ハーフラインを検出しているとき」を参照。
167 :                  default:
168 :                          break;
169 :
170 :                  }
171 :
172 :                  break;
```

通常トレースでは、初めに音を止めています。これは、クロスライン検出後のトレース、ハーフライン検出後のトレースのプログラムに分岐したときに出した音を止めるためです。switch 文では、sensor 関数の戻り値によって case 文が分岐します。case 文の内容については、以降に説明します。

(1) 中央を走行しているとき

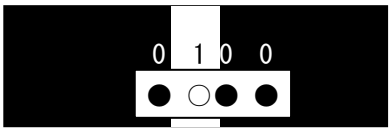
```
111 :                  case 0x06:
112 :                          // 0000 0110 センター→まっすぐ
113 :                          motor( 100, 100 );
114 :                          break;
```



センサーが“0x06”の状態です。この状態は、上図のようにラインの中央を走行している状態です。左のモーターを「100%」右のモーターを「100%」で回し、直進させます。

(2) 少し右側を走行しているとき

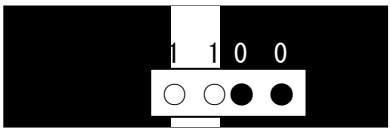
```
116 :                   case 0x04:
117 :                           // 0000 0100 少し右寄り→左へ小曲げ
118 :                           motor( 85, 100 );
119 :                           break;
```



センサーが“0x04”の状態です。この状態は、上図のようにラインの少し右側を走行している状態です。左のモーターを「85%」右のモーターを「100%」で回し、中央にセンサーが来るようにします。

(3) 中くらい右側を走行しているとき

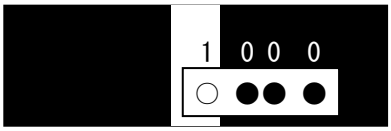
```
121 :                   case 0x0c:
122 :                           // 0000 1100 中くらい右寄り→左へ中曲げ
123 :                           motor( 70, 100 );
124 :                           break;
```



センサーが“0x0c”の状態です。この状態は、上図のようにラインの中くらい右側を走行している状態です。左のモーターを「70%」右のモーターを「100%」で回し、中央にセンサーが来るようにします。

(4) 大きく右側を走行しているとき

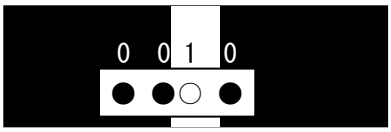
```
126 :                   case 0x08:
127 :                           // 0000 1000 大きく右寄り→左へ大曲げ
128 :                           motor( 55, 100 );
129 :                           break;
```



センサーが“0x08”の状態です。この状態は、上図のようにラインの大きく右側を走行している状態です。左のモーターを「55%」右のモーターを「100%」で回し、中央にセンサーが来るようにします。

(5) 少し左側を走行しているとき

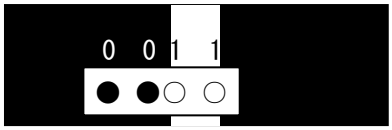
```
131 :                   case 0x02:
132 :                           // 0000 0010 少し左寄り→右へ小曲げ
133 :                           motor( 100, 85 );
134 :                           break;
```



センサーが“0x02”の状態です。この状態は、上図のようにラインの少し左側を走行している状態です。左のモーターを「100%」右のモーターを「85%」で回し、中央にセンサーが来るようにします。

(6) 中くらい左側を走行しているとき

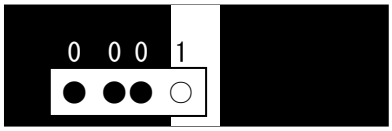
```
136 :                   case 0x03:
137 :                           // 0000 0011 中くらい左寄り→右へ中曲げ
138 :                           motor( 100, 70 );
139 :                           break;
```



センサーが“0x03”の状態です。この状態は、上図のようにラインの中くらい左側を走行している状態です。左のモーターを「100%」右のモーターを「70%」で回し、中央にセンサーが来るようにします。

(7) 大きく左側を走行しているとき

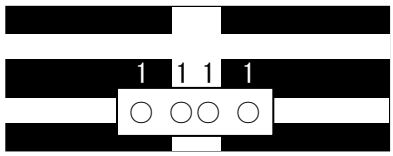
```
141 :                   case 0x01:
142 :                           // 0000 0001 大きく左寄り→右へ大曲げ
143 :                           motor( 100, 55 );
144 :                           break;
```



センサーが“0x01”の状態です。この状態は、上図のようにラインの大きく左側を走行している状態です。左のモーターを「100%」右のモーターを「55%」で回し、中央にセンサーが来るようにします。

(8) クロスラインを検出しているとき

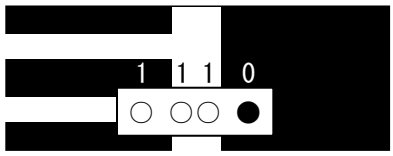
```
146 :                   case 0x0f:
147 :                           // 0000 1111 クロスライン検出
148 :                           motor( 100, 100 );
149 :                           cnt1 = 0;
150 :                           pattern = 21;
151 :                           break;
```



センサーが“0x0f”の状態です。この状態は、上図のようにクロスラインを検出している状態です。左のモーターを「100%」右のモーターを「100%」で回し、直進させます。cnt1 変数をクリアして、パターン 21 に行きます。

(9) 左ハーフラインを検出しているとき

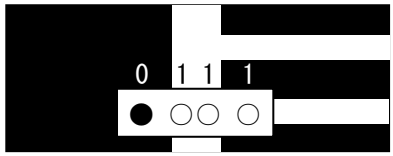
```
153 :                   case 0x0e:
154 :                           // 0000 1110 左ハーフライン検出
155 :                           motor( 100, 100 );
156 :                           cnt1 = 0;
157 :                           pattern = 31;
158 :                           break;
```



センサーが“0x0e”の状態です。この状態は、上図のように左ハーフラインを検出している状態です。左のモーターを「100%」右のモーターを「100%」で回し、直進させます。cnt1 変数をクリアして、パターン 31 に行きます。

(10) 右ハーフラインを検出しているとき

```
160 :                   case 0x07:
161 :                           // 0000 0111 右ハーフライン検出
162 :                           motor( 100, 100 );
163 :                           cnt1 = 0;
164 :                           pattern = 41;
165 :                           break;
```



センサーが“0x07”の状態です。この状態は、上図のように右ハーフラインを検出している状態です。左のモーターを「100%」右のモーターを「100%」で回し、直進させます。cnt1 変数をクリアして、パターン 41 に行きます。

5.16.6 パターン 21：クロスライン検出後のトレース、クランク検出

プログラム

```
174 :          case 21:
175 :                  // クロスライン検出後のトレース、クランク検出
176 :                  beep(Def_C3);
177 :
178 :                  switch( ( sensor() & 0x0f ) ){
179 :                  case 0x06:
180 :                          // 0000 0110 センタ→まっすぐ
181 :                          motor( 100, 100 );
182 :                          break;
183 :
184 :                  case 0x04:
185 :                          // 0000 0100 少し右寄り→左へ小曲げ
186 :                          motor( 85, 100 );
187 :                          break;
188 :
189 :                  case 0x0c:
190 :                          // 0000 1100 中くらい右寄り→左へ中曲げ
191 :                          motor( 70, 100 );
192 :                          break;
193 :
194 :                  case 0x08:
195 :                          // 0000 1000 大きく右寄り→左へ大曲げ
196 :                          motor( 55, 100 );
197 :                          break;
198 :
199 :                  case 0x02:
200 :                          // 0000 0010 少し左寄り→右へ小曲げ
201 :                          motor( 100, 85 );
202 :                          break;
203 :
204 :                  case 0x03:
205 :                          // 0000 0011 中くらい左寄り→右へ中曲げ
206 :                          motor( 100, 70 );
207 :                          break;
208 :
209 :                  case 0x01:
210 :                          // 0000 0001 大きく左寄り→右へ大曲げ
211 :                          motor( 100, 55 );
212 :                          break;
213 :
214 :                  default:
215 :                          break;
216 :
217 :                  }
219～239 :  「(1) プログラム」を参照。
241 :                  break;
```

クロスライン検出後のトレースでは、初めにドの音を出しています。これは、クロスライン検出後のトレースに入ったことが分かるようにするためです。switch 文では、sensor 関数の戻り値によって case 文が分岐します。この部分は、通常トレースと同じになっています。

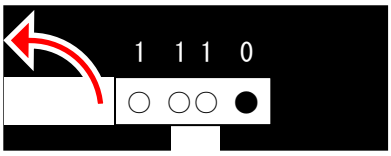
(1) プログラム

```
219 :                   if( cnt1 >= 1000 ){
220 :                           switch( ( sensor() & 0x0f ) ){
221～226 :  「(1-1) 左クランクを検出しているとき」を参照。
228～233 :  「(1-2) 右クランクを検出しているとき」を参照。
235 :                           default:
236 :                                   break;
237 :
238 :                           }
239 :                   }
```

if 文では、cnt1 変数が 1000 以上の（1000［ms］経過した）場合、{} 内の文を実行します。1000［ms］経過するまで実行しないようにしているのは、クロスライン上を走行しているときにクランクの検出をしてしまうのを避けるためです。{} 内の switch 文では、sensor 関数の戻り値によって case 文が分岐します。case 文の内容については、以降に説明します。

(1-1) 左クランクを検出しているとき

```
221 :                           case 0x0e:
222 :                                   // 0000 1110 左クランク検出
223 :                                   motor( 0, 90 );
224 :                                   cnt1 = 0;
225 :                                   pattern = 22;
226 :                                   break;
```

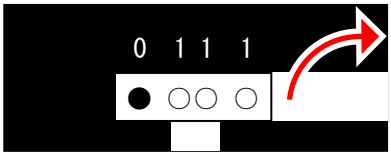


センサーが“0x0e”の状態です。この状態は、上図のように左クランクを検出している状態です。左のモーターを「n%」右のモーターを「90%」で回し、左クランクを曲がります。cnt1 変数をクリアして、パターン 22 に行きます。

「n%」の部分には任意の値を入れ、正しく曲がれるように調整をしてください。

(1-2) 右クランクを検出しているとき

```
228 :                           case 0x07:
229 :                                   // 0000 0111 右クランク検出
230 :                                   motor( 90, 0 );
231 :                                   cnt1 = 0;
232 :                                   pattern = 22;
233 :                                   break;
```



センサーが“0x07”の状態です。この状態は、上図のように右クランクを検出している状態です。左のモーターを「90%」右のモーターを「n%」で回し、右クランクを曲がります。cnt1 変数をクリアして、パターン 22 に行きます。

「n%」の部分には任意の値を入れ、正しく曲がれるように調整をしてください。

5.16.7 パターン 22：クランクの曲げ動作継続処理

プログラム

```
243 :          case 22:
244 :                  // クランクの曲げ動作継続処理
245 :                  if( cnt1 >= 1000 ){
246 :                          pattern = 11;
247 :                  }
248 :
249 :                  break;
```

if 文では、cnt1 変数が 1000 以上の（1000［ms］経過した）場合、{} 内の文を実行します。1000［ms］経過するまで実行しないようにしているのは、クランクの曲げ動作を継続させるためです。{} 内では、パターン 11 に行きます。

5.16.8 パターン 31：左ハーフライン検出後のトレース、左レーンチェンジ検出

プログラム

```
251 :          case 31:
252 :                  // 左ハーフライン検出後のトレース、左レーンチェンジ検出
253 :                  beep(Def_D3);
254 :
255 :                  switch( ( sensor() & 0x0f ) ){
256 :                  case 0x06:
257 :                          // 0000 0110 センタ→まっすぐ
258 :                          motor( 100, 100 );
259 :                          break;
260 :
261 :                  case 0x04:
262 :                          // 0000 0100 少し右寄り→左へ小曲げ
263 :                          motor( 85, 100 );
264 :                          break;
265 :
266 :                  case 0x0c:
267 :                          // 0000 1100 中くらい右寄り→左へ中曲げ
268 :                          motor( 70, 100 );
269 :                          break;
270 :
271 :                  case 0x08:
272 :                          // 0000 1000 大きく右寄り→左へ大曲げ
273 :                          motor( 55, 100 );
274 :                          break;
275 :
276 :                  case 0x02:
277 :                          // 0000 0010 少し左寄り→右へ小曲げ
278 :                          motor( 100, 85 );
279 :                          break;
280 :
281 :                  case 0x03:
282 :                          // 0000 0011 中くらい左寄り→右へ中曲げ
283 :                          motor( 100, 70 );
284 :                          break;
285 :
286 :                  case 0x01:
287 :                          // 0000 0001 大きく左寄り→右へ大曲げ
288 :                          motor( 100, 55 );
289 :                          break;
290 :
291 :                  case 0x0f:
292 :                          // 0000 1111 クロスライン検出
293 :                          motor( 100, 100 );
294 :                          cnt1 = 0;
295 :                          pattern = 21;
296 :                          break;
297 :
298 :                  default:
299 :                          break;
300 :
301 :                  }
303～316 :  「(1) プログラム」を参照。
318 :                  break;
```

左ハーフライン検出後のトレースでは、初めにレの音を出しています。これは、左ハーフライン検出後のトレースに入ったことが分かるようにするためです。switch 文では、sensor 関数の戻り値によって case 文が分岐します。この部分は、通常トレースと同じになっています。

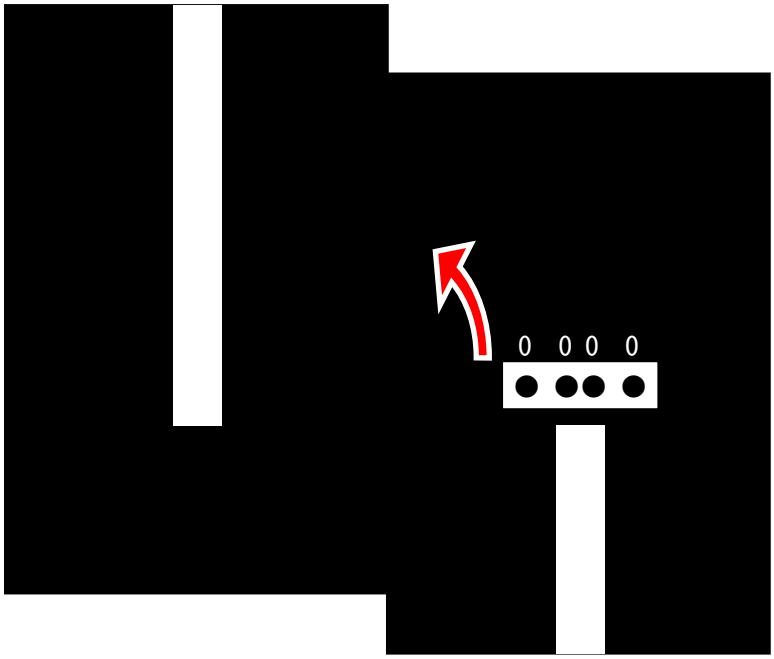
(1) プログラム

```
303 :                   if( cnt1 >= 1000 ){
304 :                           switch( ( sensor() & 0x0f ) ){
305～310 :  「(1-1) 左レーンチェンジを検出しているとき」を参照。
312 :                           default:
313 :                                   break;
314 :
315 :                           }
316 :                   }
```

if 文では、cnt1 変数が 1000 以上の（1000［ms］経過した）場合、{} 内の文を実行します。1000［ms］経過するまで実行しないようにしているのは、左ハーフライン検出後にラインから外れた場合に、左レーンチェンジの検出をしてしまうのを避けるためです。{}内の switch 文では、sensor 関数の戻り値によって case 文が分岐します。case 文の内容については、以降に説明します。

(1-1) 左レーンチェンジを検出しているとき

```
305 :                           case 0x00:
306 :                                   // 0000 0000 左レーンチェンジ検出
307 :                                   motor( 0, 100 );
308 :                                   cnt1 = 0;
309 :                                   pattern = 32;
310 :                                   break;
```



センサーが“0x00”の状態です。この状態は、上図のように左レーンチェンジを検出している状態です。左のモーターを「n%」右のモーターを「100%」で回し、左レーンチェンジを曲がります。cnt1 変数をクリアして、パターン 32 に行きます。

「n%」の部分には任意の値を入れ、正しく曲がれるように調整をしてください。

5.16.9 パターン 32：左レーンチェンジ曲げ動作継続処理

プログラム

```
320 :          case 32:
321 :                  // 左レーンチェンジ曲げ動作継続処理
322 :                  if( cnt1 >= 700 ){
323 :                          motor( 100, 100 );
324 :                          cnt1 = 0;
325 :                          pattern = 33;
326 :                  }
327 :
328 :                  break;
```

if 文では、cnt1 変数が 700 以上の（700［ms］経過した）場合、{} 内の文を実行します。
700［ms］経過するまで実行しないようにしているのは、左レーンチェンジの曲げ動作を継続させるためです。{} 内では、左のモーターを「100%」右のモーターを「100%」で回し、cnt1 変数をクリアして、パターン 33 に行きます。

5.16.10 パターン 33：左レーンチェンジ終了検出

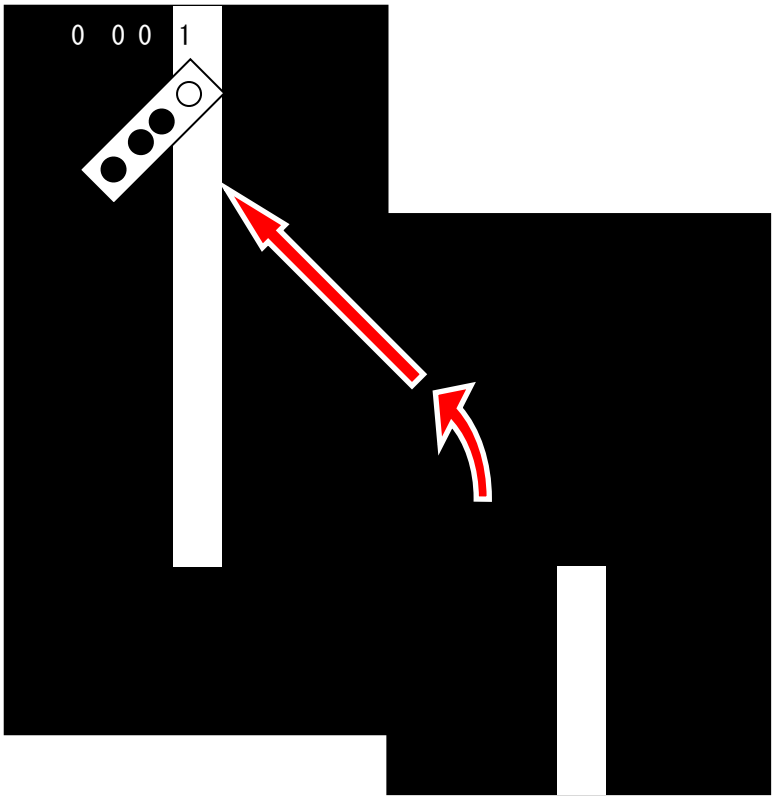
プログラム

```
330 :          case 33:
331 :                  // 左レーンチェンジ終了検出
332 :                  if( cnt1 >= 500 ){
333 :                          switch( ( sensor() & 0x0f ) ){
334～337 :  「(1) 左レーンチェンジの終了を検出しているとき」を参照。
338 :                          default:
339 :                                  break;
340 :                          }
341 :                  }
342 :
343 :                  break;
```

if 文では、cnt1 変数が 500 以上の（500［ms］経過した）場合、{} 内の文を実行します。
500［ms］経過するまで実行しないようにしているのは、左レーンチェンジ曲げ動作継続処理で、左右のモーターを同じ速度で回しているのを継続させるためです。{}内の switch 文では、sensor 関数の戻り値によって case 文が分岐します。case 文の内容については、以降に説明します。

(1) 左レーンチェンジの終了を検出しているとき

```
334 :                           case 0x01:
335 :                                   // 0000 0001 左レーンチェンジ終了検出
336 :                                   pattern = 11;
337 :                                   break;
```



センサーが“0x01”の状態です。この状態は、上図のように左レーンチェンジの終了を検出している状態です。パターン 11 に行きます。

5.16.11 パターン 41：右ハーフライン検出後のトレース、右レーンチェンジ検出

プログラム

```
345 :          case 41:
346 :                  // 右ハーフライン検出後のトレース、右レーンチェンジ検出
347 :                  beep(Def_E3);
348 :
349 :                  switch( ( sensor() & 0x0f ) ){
350 :                  case 0x06:
351 :                          // 0000 0110 センタ→まっすぐ
352 :                          motor( 100, 100 );
353 :                          break;
354 :
355 :                  case 0x04:
356 :                          // 0000 0100 少し右寄り→左へ小曲げ
357 :                          motor( 85, 100 );
358 :                          break;
359 :
360 :                  case 0x0c:
361 :                          // 0000 1100 中くらい右寄り→左へ中曲げ
362 :                          motor( 70, 100 );
363 :                          break;
364 :
365 :                  case 0x08:
366 :                          // 0000 1000 大きく右寄り→左へ大曲げ
367 :                          motor( 55, 100 );
368 :                          break;
369 :
370 :                  case 0x02:
371 :                          // 0000 0010 少し左寄り→右へ小曲げ
372 :                          motor( 100, 85 );
373 :                          break;
374 :
375 :                  case 0x03:
376 :                          // 0000 0011 中くらい左寄り→右へ中曲げ
377 :                          motor( 100, 70 );
378 :                          break;
379 :
380 :                  case 0x01:
381 :                          // 0000 0001 大きく左寄り→右へ大曲げ
382 :                          motor( 100, 55 );
383 :                          break;
384 :
385 :                  case 0x0f:
386 :                          // 0000 1111 クロスライン検出
387 :                          motor( 100, 100 );
388 :                          cnt1 = 0;
389 :                          pattern = 21;
390 :                          break;
391 :
392 :                  default:
393 :                          break;
394 :
395 :                  }
397～410 :  「(1) プログラム」を参照。
412 :                  break;
```

右ハーフライン検出後のトレースでは、初めにミの音を出しています。これは、右ハーフライン検出後のトレースに入ったことが分かるようにするためです。switch 文では、sensor 関数の戻り値によって case 文が分岐します。この部分は、通常トレースと同じになっています。

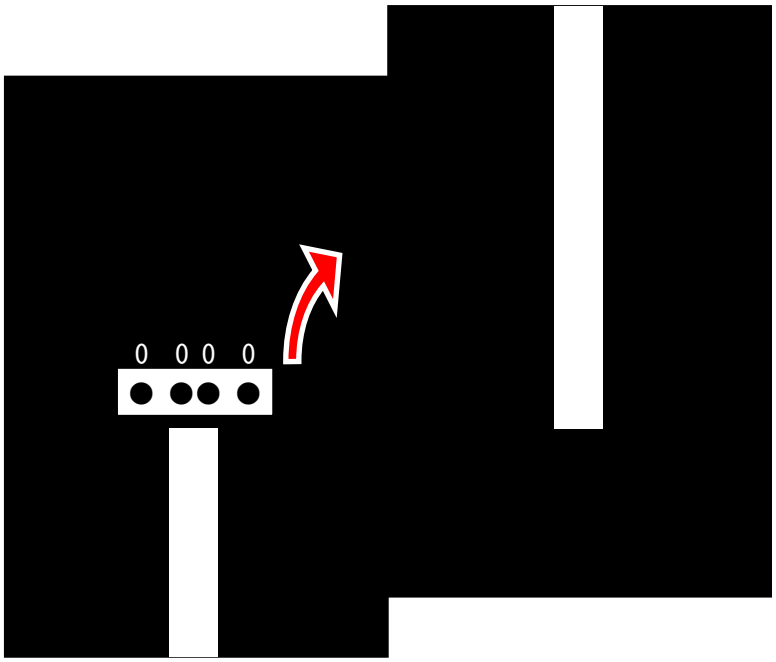
（1）プログラム

```
397 :                   if( cnt1 >= 1000 ){
398 :                           switch( ( sensor() & 0x0f ) ){
399～404 :  「(1-1) 右レーンチェンジを検出しているとき」を参照。
406 :                           default:
407 :                                   break;
408 :
409 :                           }
410 :                   }
```

if 文では、cnt1 変数が 1000 以上の（1000［ms］経過した）場合、{} 内の文を実行します。1000［ms］経過するまで実行しないようにしているのは、左ハーフライン検出後にラインから外れた場合に、左レーンチェンジの検出をしてしまうのを避けるためです。{}内の switch 文では、sensor 関数の戻り値によって case 文が分岐します。case 文の内容については、以降に説明します。

（1-1）右レーンチェンジを検出しているとき

```
399 :                           case 0x00:
400 :                                   // 0000 0000 右レーンチェンジ検出
401 :                                   motor( 100, 0 );
402 :                                   cnt1 = 0;
403 :                                   pattern = 42;
404 :                                   break;
```



センサーが“0x00”の状態です。この状態は、上図のように右レーンチェンジを検出している状態です。左のモーターを「100%」右のモーターを「n%」で回し、右レーンチェンジを曲がります。cnt1 変数をクリアして、パターン 42 に行きます。

「n%」の部分は任意の値を入れ、正しく曲がれるように調整をしてください。

5.16.12 パターン 42：右レーンチェンジ曲げ動作継続処理

プログラム

```
414 :          case 42:
415 :                  // 右レーンチェンジ曲げ動作継続処理
416 :                  if( cnt1 >= 700 ){
417 :                          motor( 100, 100 );
418 :                          cnt1 = 0;
419 :                          pattern = 43;
420 :                  }
421 :
422 :                  break;
```

if 文では、cnt1 変数が 700 以上の（700［ms］経過した）場合、{} 内の文を実行します。
700［ms］経過するまで実行しないようにしているのは、右レーンチェンジの曲げ動作を継続させるためです。{} 内では、左のモーターを「100%」右のモーターを「100%」で回し、cnt1 変数をクリアして、パターン 43 に行きます。

5.16.13 パターン 43：右レーンチェンジ終了検出

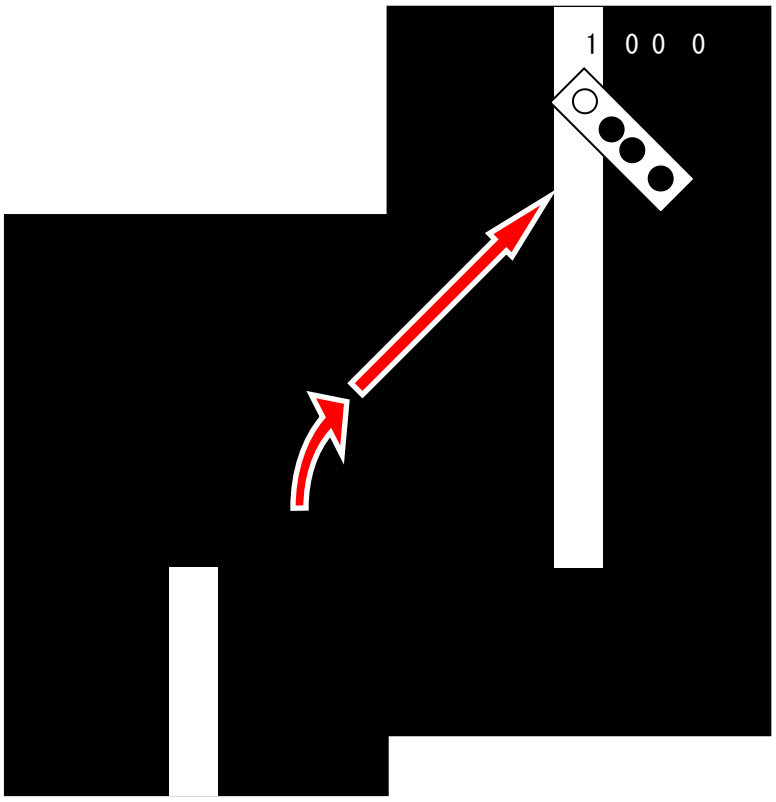
プログラム

```
424 :          case 43:
425 :                  // 右レーンチェンジ終了検出
426 :                  if( cnt1 >= 500 ){
427 :                          switch( ( sensor() & 0x0f ) ){
428～431 :  「(1) 左レーンチェンジの終了を検出しているとき」を参照。
432 :                          default:
433 :                                  break;
434 :                          }
435 :                  }
436 :
437 :                  break;
```

if 文では、cnt1 変数が 500 以上の（500［ms］経過した）場合、{} 内の文を実行します。
500［ms］経過するまで実行しないようにしているのは、右レーンチェンジ曲げ動作継続処理で、左のモーターを「100%」右のモーターを「100%」で回しているのを継続させるためです。{} 内の switch 文では、sensor 関数の戻り値によって case 文が分岐します。case 文の内容については、以降に説明します。

(1) 右レーンチェンジの終了を検出しているとき

```
428 :                       case 0x08:
429 :                               // 0000 1000 右レーンチェンジ終了検出
430 :                               pattern = 11;
431 :                               break;
```



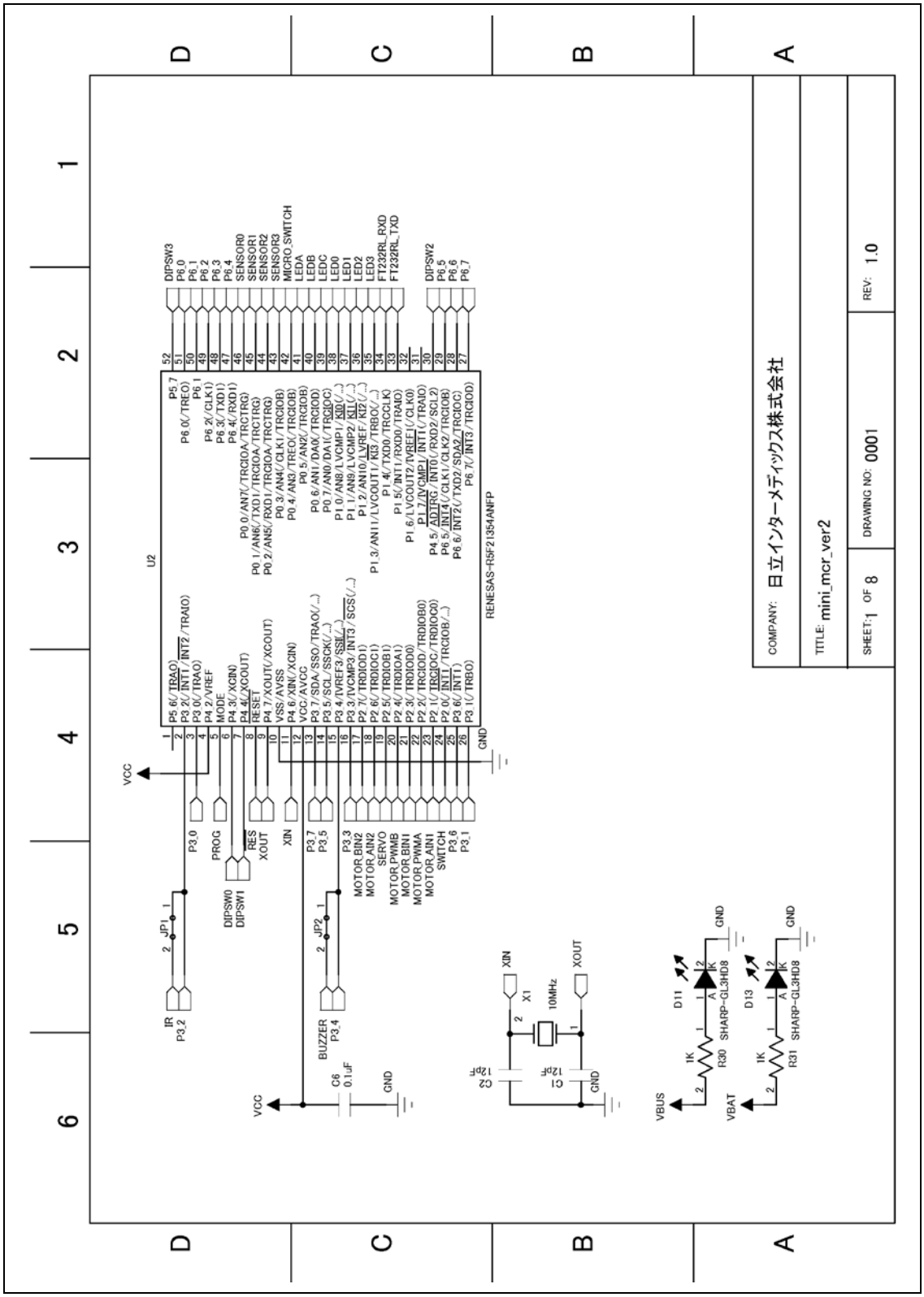
センサーが“0x08”の状態です。この状態は、上図のように右レーンチェンジの終了を検出している状態です。パターン 11 に行きます。

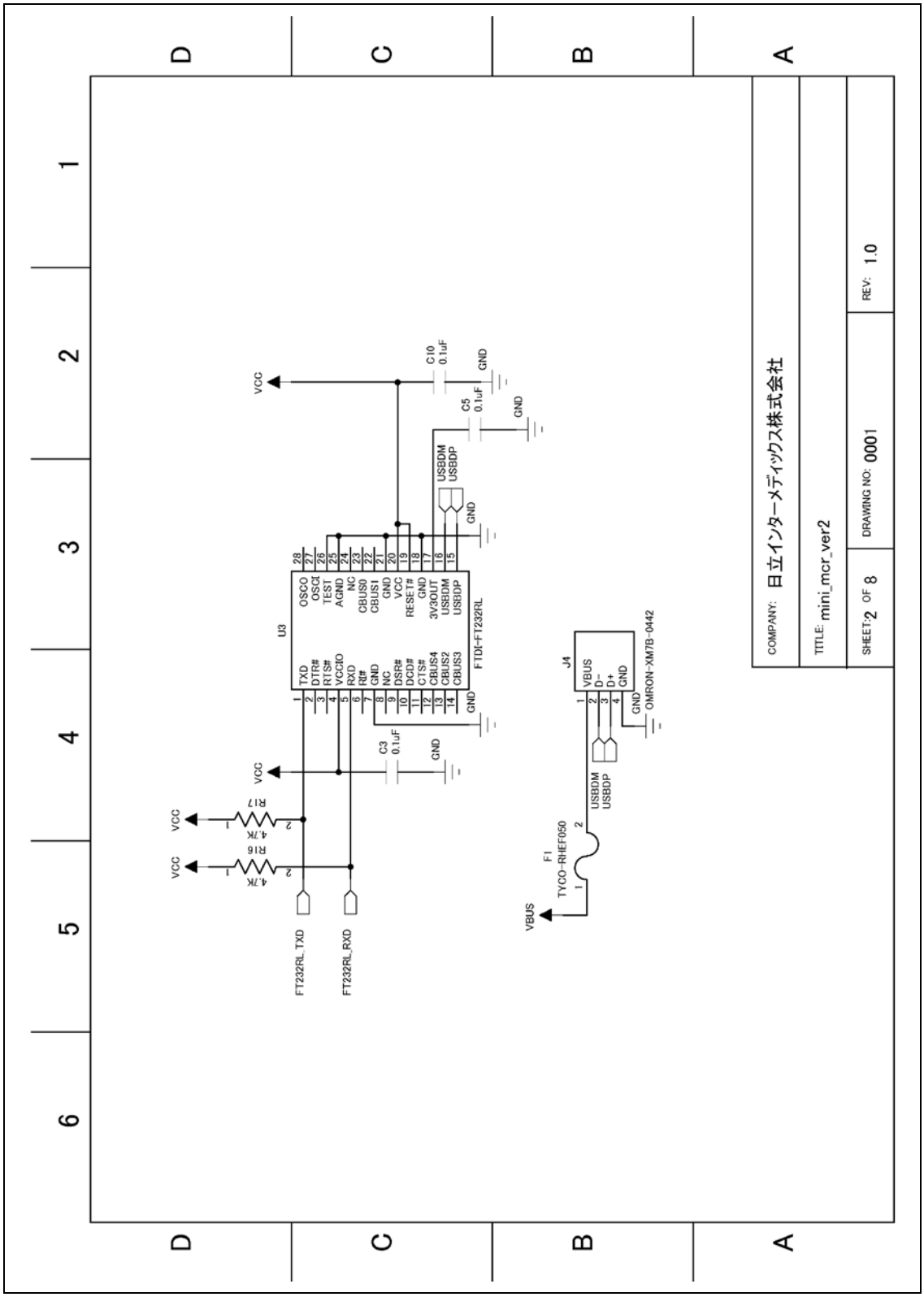
6. 仕様

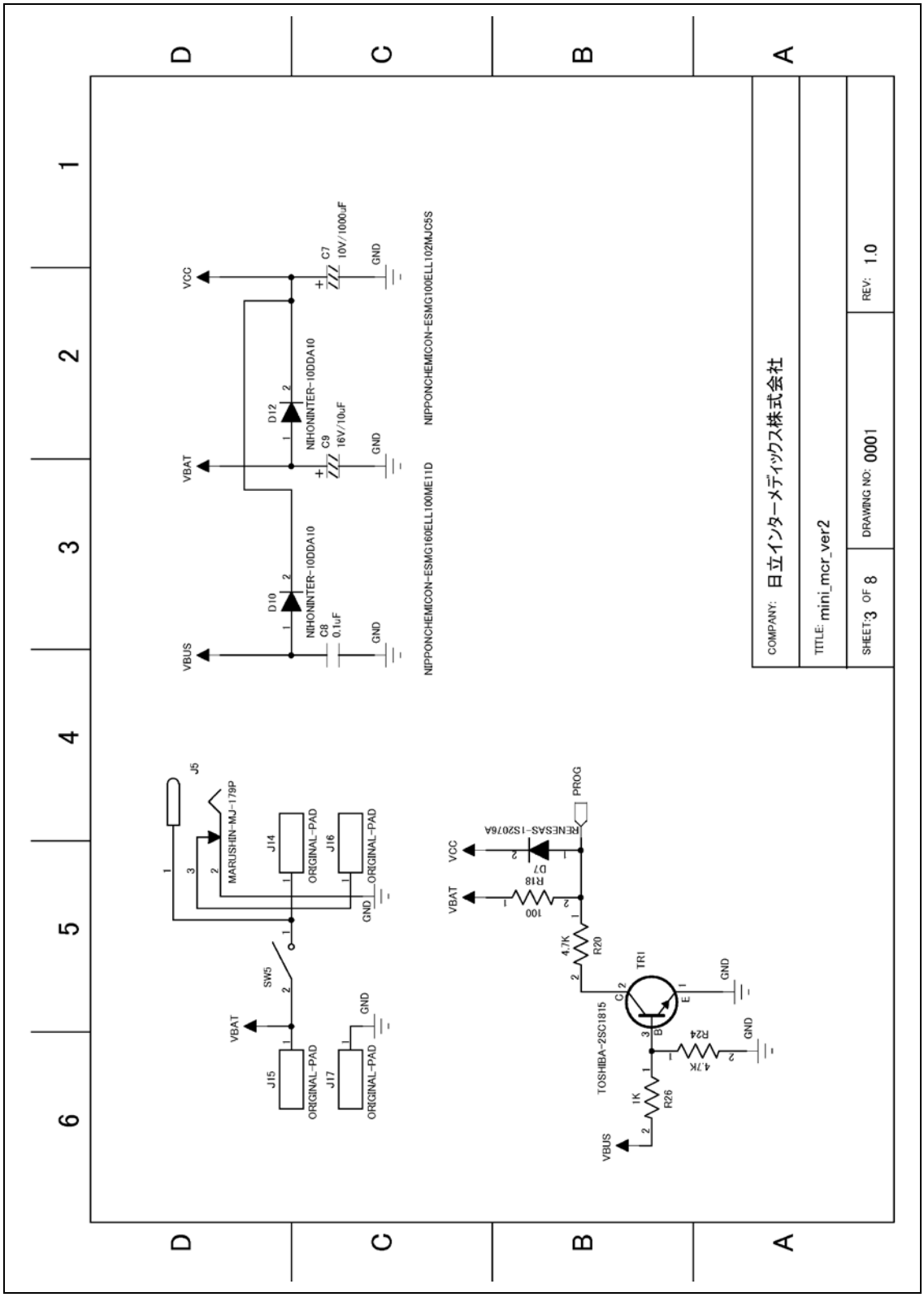
6.1 仕様

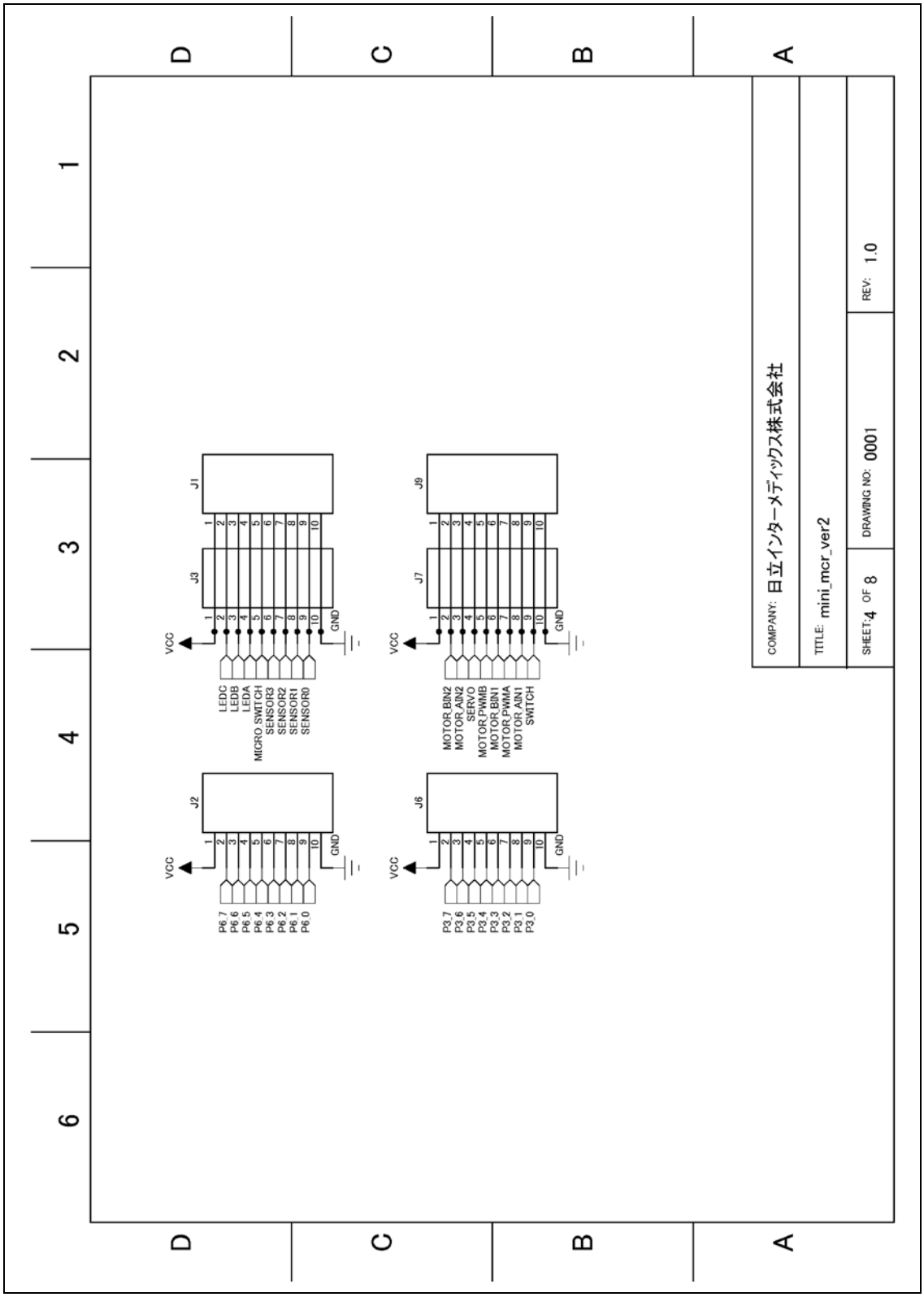
内容	詳細
マイコン	●2013 年度以降： ルネサス エレクトロニクス製　R8C/35C（R5F21356CNFP） ●2012 年度以前 ルネサス エレクトロニクス製　R8C/35A（R5F21356ANFP） ※R8C/35A と R8C/35C は、ミニマイコンカーで使う機能ではほぼ同等の機能です
電源	単 3 電池 4 本（アルカリ電池、充電電池可能） ※別売り DC ジャックコネクタと AC アダプタを使用することにより、商用電源(AC100V)での動作可能
プログラム開発	ブロックソフト､またはルネサス統合開発環境による C 言語でのプログラム開発　※各ソフトは、web サイトよりダウンロード可能
プログラム書き込み	パソコンより USB コネクタにて書き込み ※USB ケーブルは、AB タイプが接続可能
組み立て内容	電子部品の半田付け(面実装部品は実装済み)、ギヤーボックス、タイヤ
ギヤーボックス	ツインモーターギヤーボックス
モーター	FA130 モーター（ツインモーターギヤーボックス付属）×2 個
タイヤ	オフロードタイヤセット
I/O	・赤外線フォトインタラプタ（ライン検出用）×4 個 ・LED×4 個 ・DIP スイッチ（4bit）×1 個 ・タクトスイッチ×1 個 ・圧電サウンダ×1 個 ・DC モータードライバ（2ch）×1 個 ・マイクロスイッチ（障害物検出用）×1 個 ・赤外線リモコン受光モジュール×1 個 ・サーボコネクタ×1 個 ・拡張 I/O コネクタ×4 個
その他	基板のセンサー部分、モータードライバ部分を分離して、マイコンボードとして使用可能

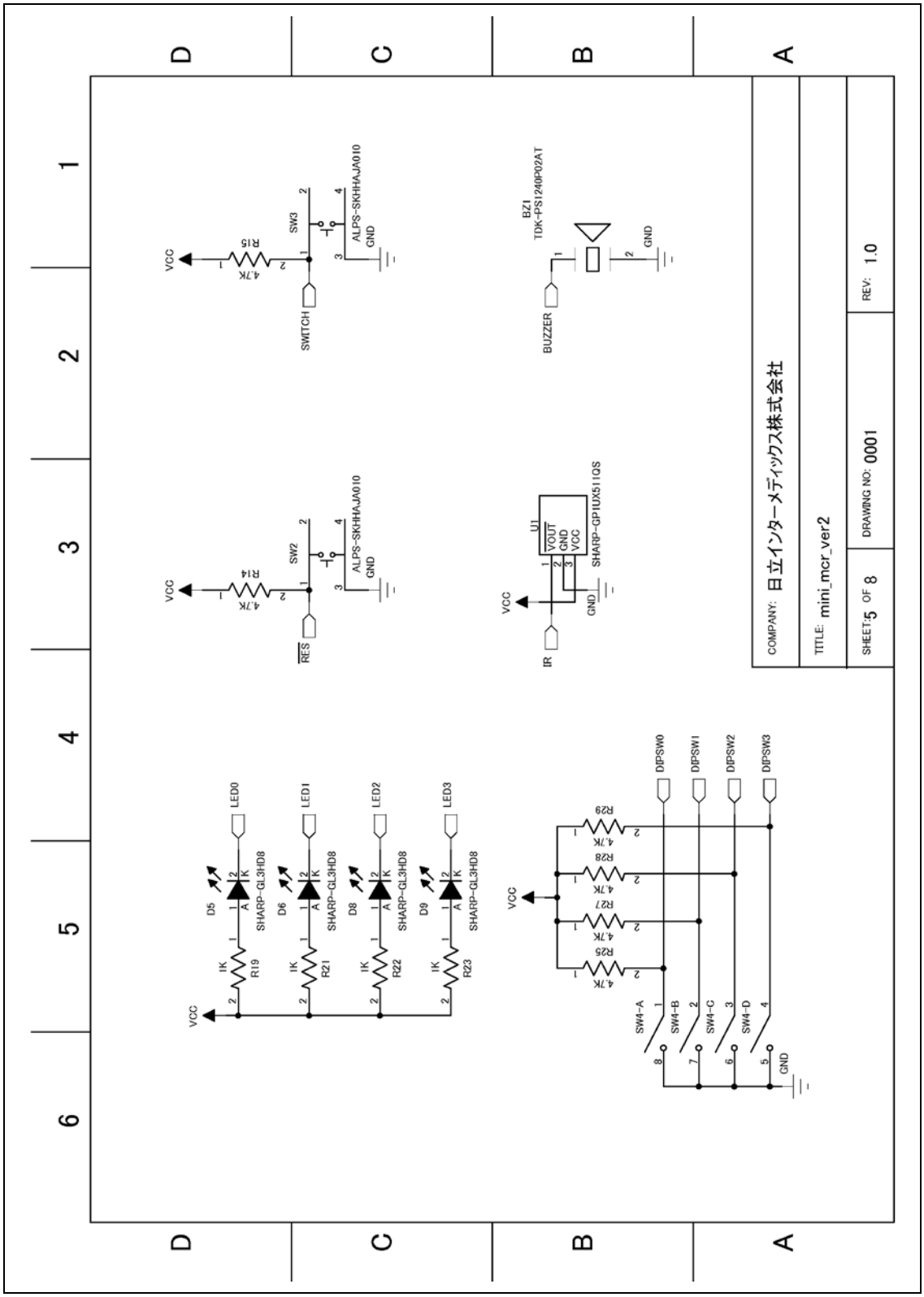
6.2 回路図

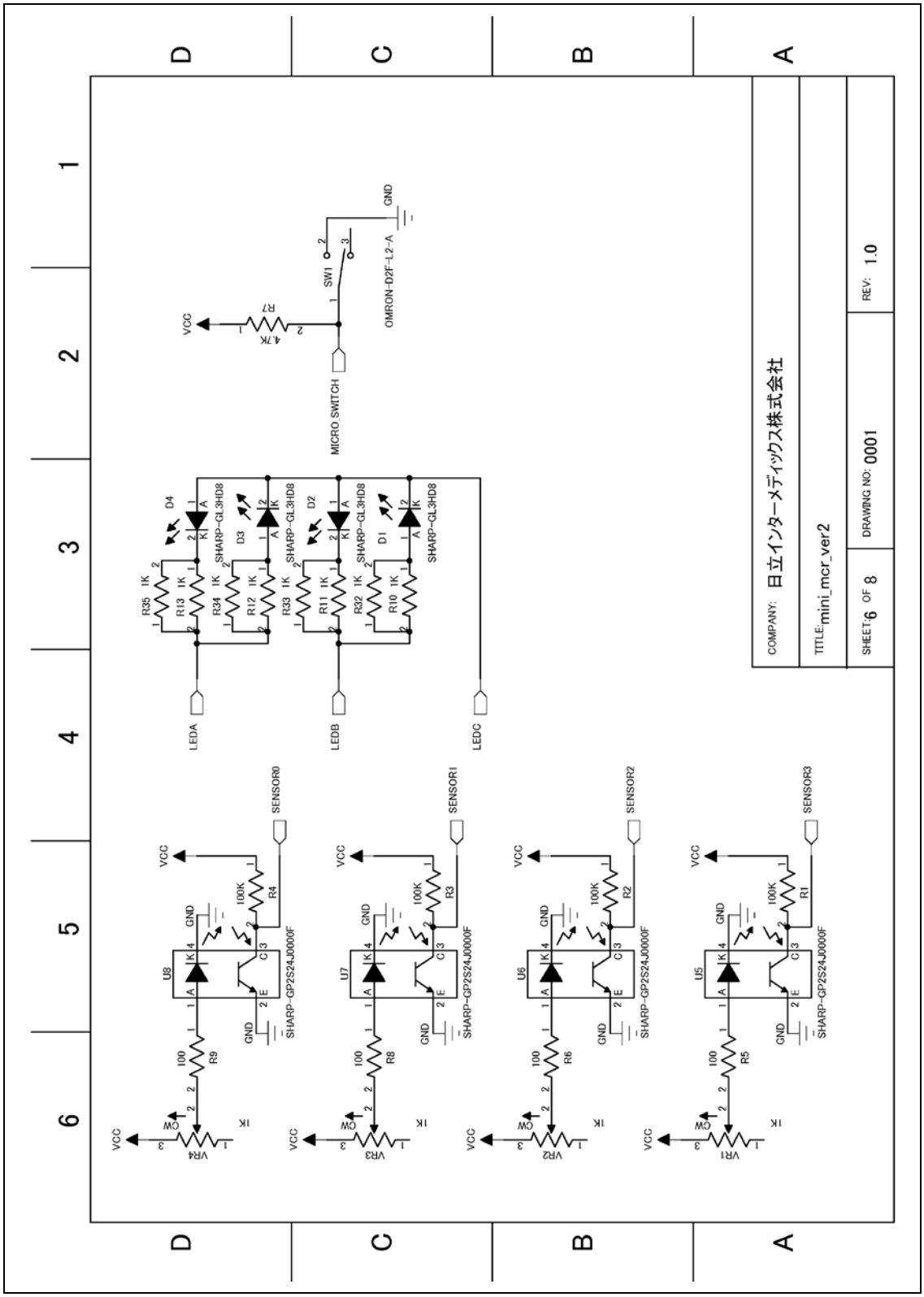

COMPANY: 日立インターメディックス株式会社
TITLE: mini_mcr_ver2
SHEET: 1 OF 8
DRAWING NO: 0001
REV: 1.0
U2
RENESAS-R5F21354ANFP
VCC
GND
XIN
XOUT
10MHz
X1
C1 12pF
C2 12pF
C6 0.1μF
JP1
JP2
IR P3.2
BUZZER P3.4
DIPSW0
DIPSW1
PROG
RES
XOUT
MOTOR_BIN2
MOTOR_AIN2
SERVO
MOTOR_PWMB
MOTOR_BIN1
MOTOR_PWMA
MOTOR_AIN1
SWITCH
VBUS
VBAT
D11
D13
1K
R30
R31
SHARP-GL3HD8
DIPSW3
DIPSW2
MICRO_SWITCH
LED3
FT232RL_RXD
FT232RL_TXD

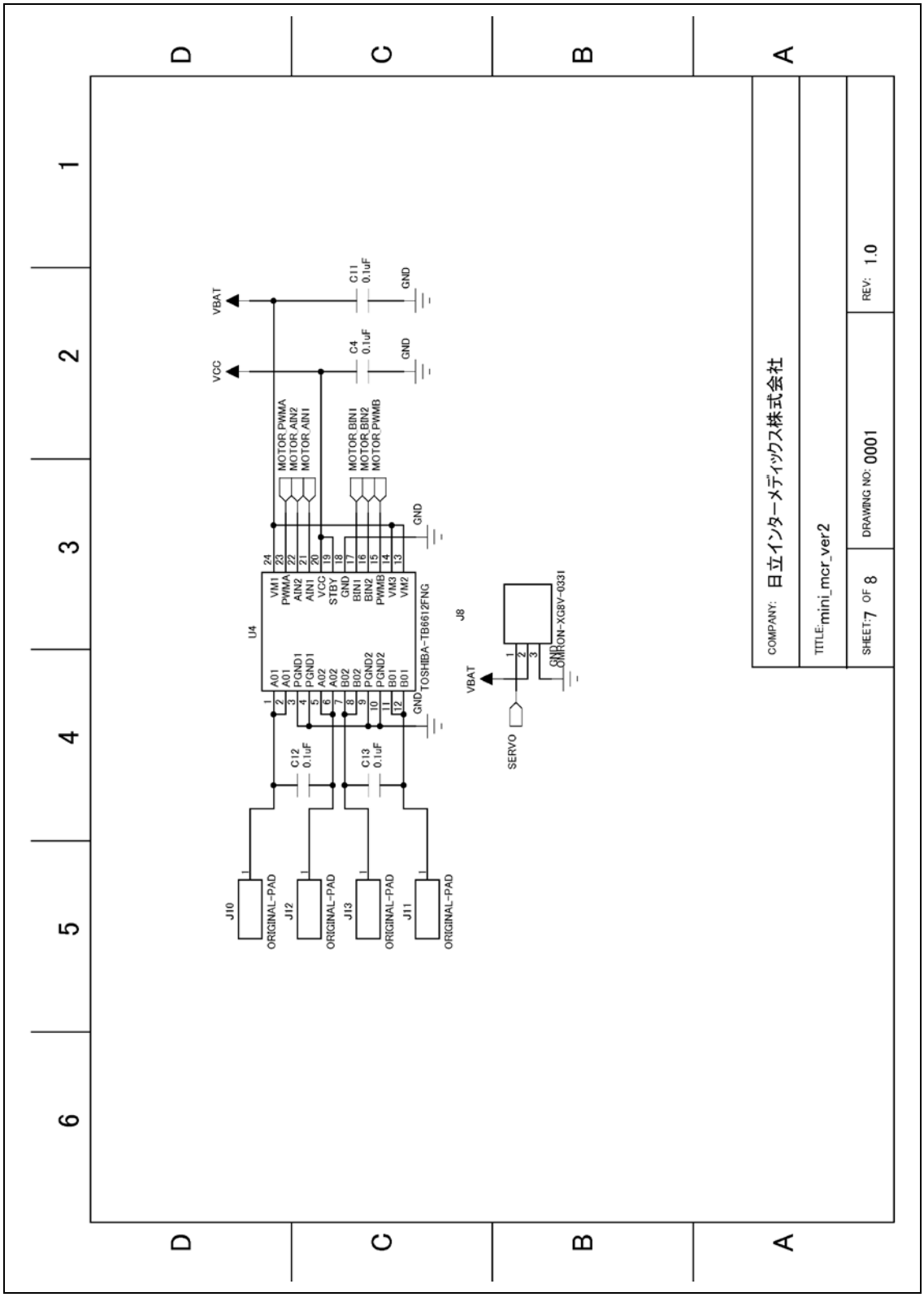

COMPANY: 日立インターメディックス株式会社
TITLE: mini_mcr_ver2
SHEET: 2 OF 8
DRAWING NO: 0001
REV: 1.0
U3
FTDI-FT232RL
J4
OMRON-XM7B-0442
F1
TYCO-RHEF050
C10
C5
C3
R17
R16
VCC
GND
USBDM
USBDP
VBUS
FT232RL_TXD
FT232RL_RXD


COMPANY: 日立インターメディックス株式会社
TITLE: mini_mcr_ver2
SHEET: 3 OF 8
DRAWING NO: 0001
REV: 1.0
VCC
VBAT
VBUS
GND
C7
10V/1000uF
C9
16V/10uF
C8
0.1uF
D12
D10
NIHONINTER-10DDA10
NIPPONCHEMICON-ESMG160ELL100ME11D
NIPPONCHEMICON-ESMG100ELL102MJC5S
J5
MARUSHIN-MJ-179P
J14
J16
J15
J17
ORGINAL-PAD
SW5
PROG
D7
RENESAS-1S2076A
R18
100
R20
4.7K
TR1
TOSHIBA-2SC1815
R24
4.7K
R26
1K


J1
J3
LEDC
LEDB
LEDA
MICRO SWITCH
SENSOR3
SENSOR2
SENSOR1
SENSOR0
VCC
GND
J9
J7
MOTOR_BIN2
MOTOR_AIN2
SERVO
MOTOR_PWMB
MOTOR_BIN1
MOTOR_PWMA
MOTOR_AIN1
SWITCH
J2
P6.7
P6.6
P6.5
P6.4
P6.3
P6.2
P6.1
P6.0
J6
P3.7
P3.6
P3.5
P3.4
P3.3
P3.2
P3.1
P3.0
COMPANY: 日立インターメディックス株式会社
TITLE: mini_mcr_ver2
SHEET: 4 OF 8
DRAWING NO: 0001
REV: 1.0


COMPANY: 日立インターメディックス株式会社
TITLE: mini_mcr_ver2
SHEET: 5 OF 8
DRAWING NO: 0001
REV: 1.0
SWITCH
ALPS-SKHHAJA010
BUZZER
TDK-PS1240P02AT
SHARP-GP1UX511QS
LED0
LED1
LED2
LED3
SHARP-GL3HD8
DIPSW0
DIPSW1
DIPSW2
DIPSW3
VCC
GND


COMPANY: 日立インターメディックス株式会社
TITLE: mini_mcr_ver2
SHEET: 6 OF 8
DRAWING NO: 0001
REV: 1.0
MICRO_SWITCH
OMRON-D2F-L2-A
SHARP-GL3HD8
SHARP-GP2S24J0000F
LEDA
LEDB
LEDC
SENSOR0
SENSOR1
SENSOR2
SENSOR3
VCC
GND


COMPANY: 日立インターメディックス株式会社
TITLE: mini_mcr_ver2
SHEET: 7 OF 8
DRAWING NO: 0001
REV: 1.0
U4
TOSHIBA-TB6612FNG
J8
OMRON-XG8V-0331
SERVO
VBAT
VCC
GND
C11 0.1uF
C4 0.1uF
C12 0.1uF
C13 0.1uF
J10
J12
J13
J11
ORIGINAL-PAD
MOTOR PWMA
MOTOR AIN2
MOTOR AIN1
MOTOR BIN1
MOTOR BIN2
MOTOR PWMB

6.3 ポート表

コネクタ	番号	端子名	接続先
J3	1		VCC
	2	P0_7/AN0/DA1(/TRCIOC)	LEDC (P0_7) ※
	3	P0_6/AN1/DA0(/TRCIOD)	LEDB (P0_6) ※
	4	P0_5/AN2(/TRCIOB)	LEDA (P0_5) ※
	5	P0_4/AN3/TREO(/TRCIOB)	マイクロスイッチ (P0_4) ※
	6	P0_3/AN4(/CLK1/TRCIOB)	赤外線フォトインタラプタ 3 (P0_3) ※
	7	P0_2/AN5(/RXD1/TRCIOA/TRCTRG)	赤外線フォトインタラプタ 2 (P0_2) ※
	8	P0_1/AN6(/TXD1/TRCIOA/TRCTRG)	赤外線フォトインタラプタ 1 (P0_1) ※
	9	P0_0/AN7(/TRCIOA/TRCTRG)	赤外線フォトインタラプタ 0 (P0_0) ※
	10		GND

※基板のセンサー部分を分離することで、J3 コネクタの信号を自由に使用できます。

コネクタ	番号	端子名	接続先
		P1_7/IVCMP1/INT1(/TRAIO)	
		P1_6/LVCOUT2/IVREF1(/CLK0)	
		P1_5(/INT1/RXD0/TRAIO)	RxD0
		P1_4(/TXD0/TRCCLK)	TxD0
		P1_3/AN11/LVCOUT1/K13/TRB0(/TRCIOC)	LED3 (P1_3)
		P1_2/AN10/LVREF/K12(/TRCIOB)	LED2 (P1_2)
		P1_1/AN9/LVCMP2/KI1(/TRCIOA/TRCTRG)	LED1 (P1_1)
		P1_0/AN8/LVCMP1/KI0(/TRCIOD)	LED0 (P1_0)

コネクタ	番号	端子名	接続先
J7	1		VCC
	2	P2_7(/TRDIOD1)	モーター右 2 (P2_7) ※
	3	P2_6(/TRDIOC1)	モーター左 2 (P2_6) ※
	4	P2_5(/TRDIOB1)	サーボ (TRDIOB1) ※
	5	P2_4(/TRDIOA1)	モーター右 PWM (TRDIOA1) ※
	6	P2_3(/TRDIOD0)	モーター右 1 (P2_3) ※
	7	P2_2(/TRCIOD/TRDIOB0)	モーター左 PWM (TRDIOB0) ※
	8	P2_1(/TRCIOC/TRDIOC0)	モーター左 1 (P2_1) ※
	9	P2_0(/INT1/TRCIOB/TRDIOA0/TRDCLK)	タクトスイッチ (P2_0)
	10		GND

※基板のモータードライバ部分を分離することで、J7 コネクタの信号を自由に使用できます。

コネクタ	番号	端子名	接続先
J6	1		VCC
	2	P3_7/SDA/SSO/TRAO(/RXD2/SCL2/TXD2/SDA2)	
	3	P3_6(/INT1)	
	4	P3_5/SCL/SSCK(/CLK2/TRCIOD)	
	5	P3_4/IVREF3/SSI(/RXD2/SCL2/TXD2/SDA2/TRCIOC)	圧電サウンダ (TRCIOC)
	6	P3_3/IVCMP3/INT3/SCS(/CTS2/RTS2/TRCCLK)	
	7	P3_2(/INT1/INT2/TRAIO)	赤外線リモコン受光モジュール (TRAIO)
	8	P3_1(/TRBO)	
	9	P3_0(/TRAO)	
	10		GND

コネクタ	番号	端子名	接続先
		P4_7/XOUT	クリスタル（XOUT）
		P4_6/XIN	クリスタル（XIN）
		P4_5/ADTRG/INT0(/RXD2/SCL2)	DIP スイッチ 2
		P4_4(/XCOUT)	DIP スイッチ 1
		P4_3(/XCIN)	DIP スイッチ 0
		P4_2/VREF	VCC

コネクタ	番号	端子名	接続先
		P5_7	DIP スイッチ 3
		P5_6(/TRAO)	

コネクタ	番号	端子名	接続先
J2	1		VCC
	2	P6_7(/INT3/TRCIOD)	
	3	P6_6/INT2(/TXD2/SDA2/TRCIOC)	
	4	P6_5/INT4(/CLK1/CLK2/TRCIOB)	
	5	P6_4(/RXD1)	
	6	P6_3(/TXD1)	
	7	P6_2(/CLK1)	
	8	P6_1	
	9	P6_0(/TREO)	
	10		GND

6.4 ピン配置図

コネクタ