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

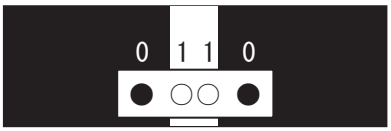
プログラム

```
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

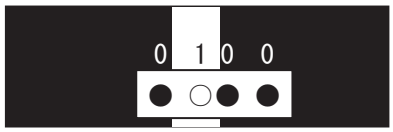
```
111 :                  case 0x06:
112 :                          // 0000 0110 センター→まっすぐ
113 :                          motor( 100, 100 );
114 :                          break;
```



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

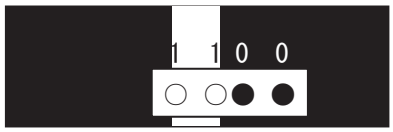
```
116 :                   case 0x04:
117 :                           // 0000 0100 少し右寄り→左へ小曲げ
118 :                           motor( 85, 100 );
119 :                           break;
```



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

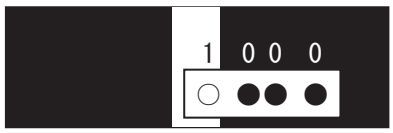
```
121 :                   case 0x0c:
122 :                           // 0000 1100 中くらい右寄り→左へ中曲げ
123 :                           motor( 70, 100 );
124 :                           break;
```



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

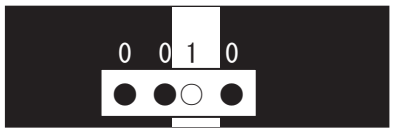
```
126 :                   case 0x08:
127 :                           // 0000 1000 大きく右寄り→左へ大曲げ
128 :                           motor( 55, 100 );
129 :                           break;
```



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

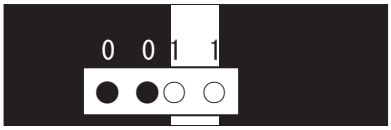
```
131 :                   case 0x02:
132 :                           // 0000 0010 少し左寄り→右へ小曲げ
133 :                           motor( 100, 85 );
134 :                           break;
```



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

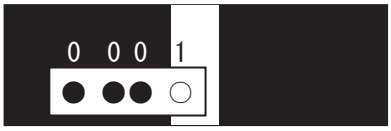
```
136 :                   case 0x03:
137 :                           // 0000 0011 中くらい左寄り→右へ中曲げ
138 :                           motor( 100, 70 );
139 :                           break;
```



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

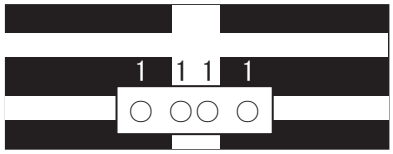
```
141 :                   case 0x01:
142 :                           // 0000 0001 大きく左寄り→右へ大曲げ
143 :                           motor( 100, 55 );
144 :                           break;
```



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

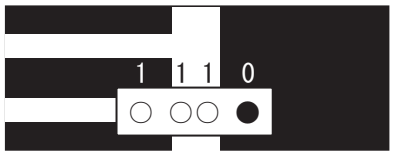
```
146 :                   case 0x0f:
147 :                           // 0000 1111 クロスライン検出
148 :                           motor( 100, 100 );
149 :                           cnt1 = 0;
150 :                           pattern = 21;
151 :                           break;
```



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

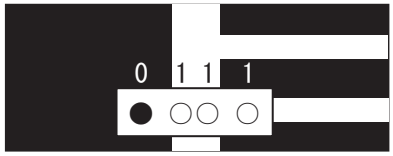
```
153 :                   case 0x0e:
154 :                           // 0000 1110 左ハーフライン検出
155 :                           motor( 100, 100 );
156 :                           cnt1 = 0;
157 :                           pattern = 31;
158 :                           break;
```



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。

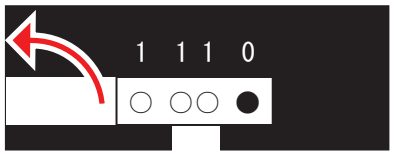
(1) プログラム

```
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;
```

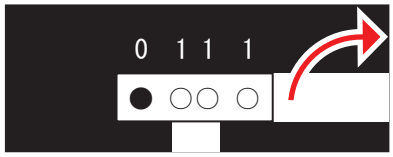


センサーが“0x0e”の状態です。この状態は、上図のように左クランクを検出している状態です。左のモーターを「n%」右のモーターを「90%」で回し、左クランクを曲がります。cnt1 変数をクリアして、パターン 22 に行きます。

「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。

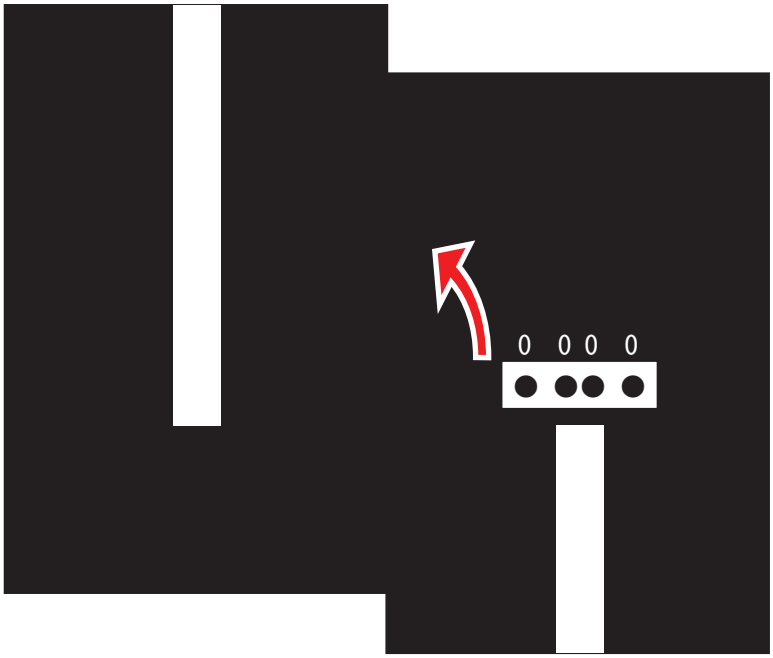
（1）プログラム

```
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

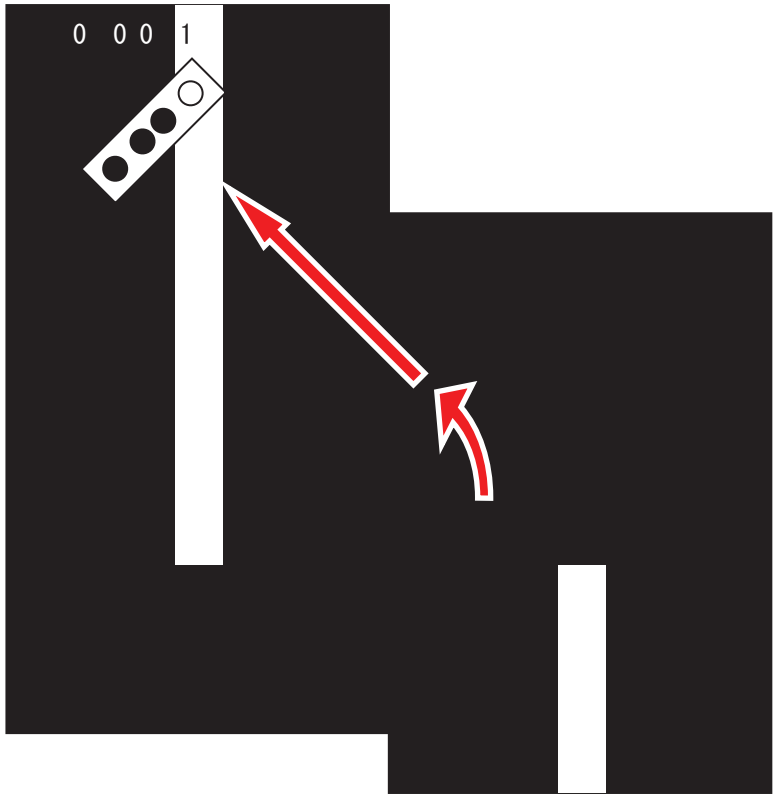
プログラム

```
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。

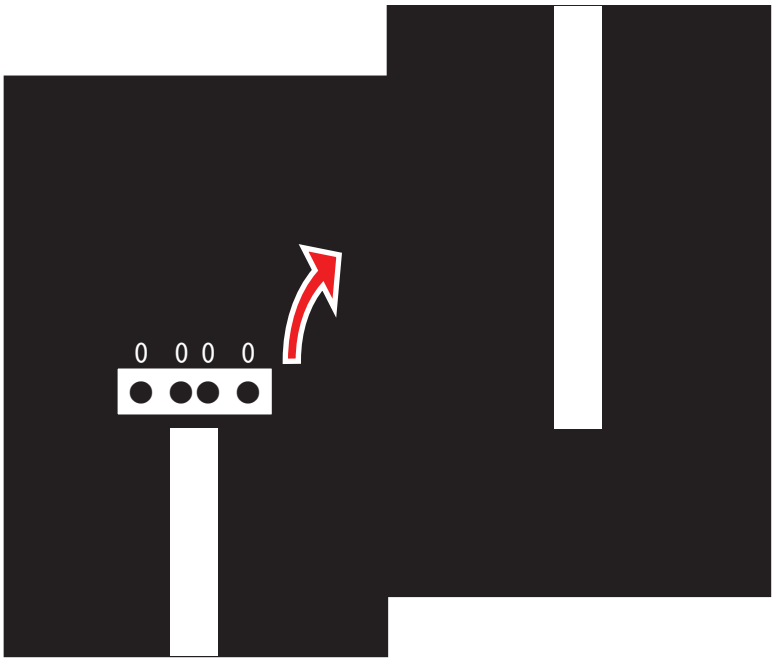
（1）プログラム

```
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

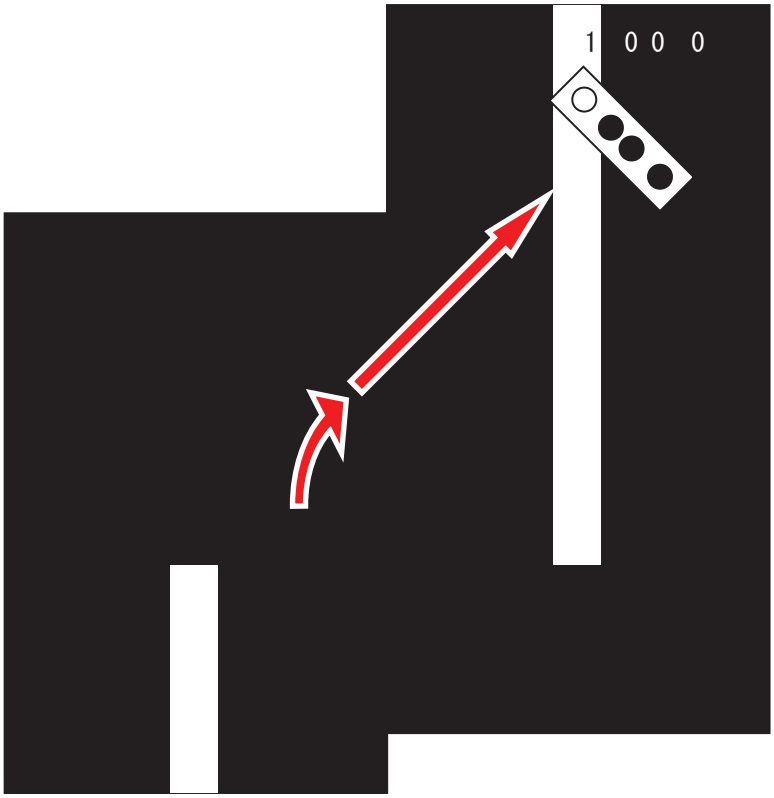
プログラム

```
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。