```
/*=======================================*/
/* インクルード                          */
/*=======================================*/
#include <stdio.h>
#include "sfr_r838a.h"                  /* R8C/38A SFR の定義ファイル      */
#include "types3_beep.h"                /* ブザー追加                      */

/*=======================================*/
/* シンボル定義                          */
/*=======================================*/
/* 定数設定 */
#define     TRC_MOTOR_CYCLE     20000   /* 左前,右前モータ PWM の周期      */
                                        /* 50[ns] * 20000 = 1.00[ms]       */
#define     TRD_MOTOR_CYCLE     20000   /* 左後,右後,ｻｰﾎﾞ ﾓｰﾀ PWM の周期   */
                                        /* 50[ns] * 20000 = 1.00[ms]       */
#define     SERVO_STEP          5       /* 1°あたりの数値                  */
#define     SERVO_CENTER        500     /* 中心                            */

#define     FREE                1       /* モータモード　フリー            */
#define     BRAKE               0       /* モータモード　ブレーキ          */
#define     LEFT                1       /* クランク左                      */
#define     RIGHT               2       /* クランク右                      */
#define     TESTSPEED           25      //テスト用の速さ（一律）

/*=======================================*/
/* プロトタイプ宣言                      */
/*=======================================*/
void init( void );
unsigned char sensor_inp( void );
unsigned char center_inp( void );
unsigned char startbar_get( void );
unsigned char dipsw_get( void );
unsigned char dipsw_get2( void );
unsigned char dipsw_get3( void );
unsigned char dipsw_get4( void );
unsigned char dipsw_getf( void );
unsigned char pushsw_get( void );
unsigned char cn6_get( void );
void led_out( unsigned char led );
void motor_r( int accele_l2, int accele_r2 );
void motor2_r( int accele_l, int accele_r );
void motor_f( int accele_l2, int accele_r2 );
void motor2_f( int accele_l, int accele_r );
void motor_mode_r( int mode_l, int mode_r );
void motor_mode_f( int mode_l, int mode_r );
void servoPwmOut( int pwm );
```

```
int check_crossline( void );
int check_zlineR( void );
int check_zlineL( void );
int check_Noline( void );
int getServoAngle( void );
int getAnalogSensor( void );
void servoControl( void );
int diff( int pwm );
void handle( int iSetAngle );

/*=======================================*/
/*  グローバル変数の宣言                 */
/*=======================================*/
int             pattern = 0;            /*  マイコンカー動作パターン      */
int             crank_mode;              /* 1:クランクモード  0:通常       */
unsigned long   cnt1 = 0;               /*  タイマ用                      */
int             h;

/*  エンコーダ関連  */
int             iTimer10;               /* 10ms カウント用                */
unsigned long   lEncoderTotal;          /*  積算値保存用                  */
unsigned long   lEncoderCrank;          /*クランク時の距離   lEncoderTotal - lEncoderCrank */
int             iEncoder;               /* 10ms 毎の最新値                */
unsigned int    uEncoderBuff;           /*  計算用   割り込み内で使用     */

/*   サーボ関連  */
int             iSensorBefore;          /*  前回のセンサ値保存            */
int             iServoPwm;               /*  サーボＰＷＭ値               */
int             iAngle0;                /*  中心時の A/D 値保存           */
int             iAngle;                 /*  角度                          */
/*  センサ関連  */
int             iSensorPattern;         /*  センサ状態保持用              */

/* TRC レジスタのバッファ  */
unsigned int    trcgrb_buff;            /* TRCGRB のバッファ              */
unsigned int    trcgrd_buff;            /* TRCGRD のバッファ              */
unsigned int    trcgrc_buff;

/*  モータドライブ基板 TypeS Ver.3 上の LED、ディップスイッチ制御  */
unsigned char   types_led;              /* LED 値設定                     */
unsigned char   types_dipsw;            /*  ディップスイッチ値保存        */

/*  レーンチェンジ  */
char Lane_Change ;                       /* 1 左  0 右                    */

/*モータ関係          */
```

```
unsigned char M_FreeMoter;

/* サーボ関係 2 */
int iSetAngle;        /* 設定したい角度(AD 値) */
int iAngleBefore2; /* 前回の角度保存 */
int iServoPwm2;       /* サーボＰＷＭ値 */

int gain;     //強さ調整

/* 速度調整 */
unsigned char     lanechangespeed;
unsigned char     cornerspeed;
unsigned char     crankspeed;
unsigned char     streetspeed;

/* コースアウト */
unsigned long causeout;

int acceleFree;



/* 内輪差値計算用　各マイコンカーに合わせて再計算して下さい */
const int revolution_difference[] = {     /* 角度から内輪、外輪回転差計算 */
      100, 98, 97, 95, 94,
      92, 91, 89, 88, 87,
      85, 84, 82, 81, 80,
      78, 77, 76, 74, 73,
      72, 70, 69, 68, 66,
      65, 64, 62, 61, 60,
      58, 57, 56, 54, 53,
      52, 50, 49, 48, 46,
      45, 43, 42, 40, 39,
      38 };

const int dipsw1_pattern[] = { 9, 10, 11, 12, 13, 14, 15, 16};
const int dipsw2_pattern[] = { 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25};
const int dipsw3_pattern[] = { 10, 11, 12, 13};
const int dipsw4_pattern[] = { 5, 6, 7, 8};
const int dipswf_pattern[] = { 0, 1};
/************************************************************************/
/* メインプログラム                                                     */
/************************************************************************/
void main( void ){
      int i;

      /* マイコン機能の初期化 */
```

```
    init();                                 /* 初期化                          */
    asm(" fset I ");                        /* 全体の割り込み許可              */
    initBeepS();                            /* ブザー関連処理                  */

    /* マイコンカーの状態初期化 */
    motor_mode_f( BRAKE, BRAKE );   //BRAKE
    motor_mode_r( BRAKE, BRAKE );
    motor_f( 0, 0 );
    motor_r( 0, 0 );
    servoPwmOut( 0 );
    setBeepPatternS( 0x8000 );




    while( 1 ) {

        //メニュー以外
        if(pattern != 1 && pattern != 2 && pattern != 3 && pattern != 4 && pattern != 5 && pattern != 6 && pattern != 7 && pattern != 8 && pattern != 9 && pattern != 10){
        //通常トレースでのコースアウト
        if(pattern != 32 &&         pattern != 33 && pattern != 42 && pattern != 43 && pattern != 61 && pattern != 62 && pattern != 63 && pattern != 71 && pattern != 72 && pattern != 73){

        if((lEncoderTotal - causeout) >= 1000){
        pattern = 2;
            }
        }
        //アクションでのコースアウト
        else if(pattern == 32 || pattern == 33 || pattern == 42 || pattern == 43 || pattern == 61 || pattern == 62 || pattern == 63 || pattern == 71 || pattern == 72 || pattern == 73){

        if((lEncoderTotal - causeout) >= 3500){
        pattern = 2;
            }
         }
        }

        if(pattern == 2){
            setBeepPatternS( 0xcc00 );
         }

    switch( pattern ) {
    case 0:
        /* プッシュスイッチ押下待ち */
                gain = 0;
```

```
    servoPwmOut( 0 );
            lEncoderTotal = 0;
            lEncoderCrank = 0;
            causeout = 0;
            iAngleBefore2 = 0;
            M_FreeMoter = 0;;
            streetspeed = (dipsw2_pattern[dipsw_get2()]+5); //直線
            cornerspeed = dipsw2_pattern[dipsw_get2()];   //曲線
            lanechangespeed = dipsw3_pattern[dipsw_get3()]; //車線変更
            crankspeed = dipsw4_pattern[dipsw_get4()];//クランク

            if( pushsw_get() && cnt1 >= 100){
                    M_FreeMoter = dipswf_pattern[dipsw_getf()];   //1 でフリーモード   0 で通常
      setBeepPatternS( 0xcc00 );
                    cnt1 = 0;
      pattern = 1;
    break;
            }
            i =   (cnt1/200) % 2 + 1;
    led_out( i );                       /* LED 点滅処理                    */
    break;

case 1:
    /* スタートバー開待ち */
            gain = 10;
    servoPwmOut( iServoPwm / 2 );
            causeout = lEncoderTotal;
    if( !startbar_get() ) {
      iAngle0 = getServoAngle();   /* 0 度の位置記憶                    */
      led_out( 0x0 );
                    cnt1 = 0;
      pattern = 11;      //11 通常走行 3 ステアリング確認 4 モーター速度
     break;
    }
    led_out( 1 << (cnt1/50) % 4 );
    break;

 case 2:      //停止
          gain = 10;
     servoPwmOut( 0 );
              motor_f( 0, 0 );
     motor_r( 0, 0 );
              led_out( 0xaa);
            break;

     case 3:      //ステアリング
```

```
  gain = 10;
        servoPwmOut( iServoPwm );
     if(iEncoder < 5){
        motor_f( 20, 20 );
  motor_r( 20, 20 );
              }
     else{
              motor_f( 0, 0 );
  motor_r( 0, 0 );
              }
            break;

  case 4:     //モータ速度
        gain = 0;
        servoPwmOut( 0 );
   if (iEncoder < 10){
        motor_f( -50, -50 );
        motor_r(-50, -50 );
                }
                else{
                        motor_f( 0, 0 );
        motor_r( 0, 0 );
                }
                break;

  case 5:   //ハンドル角度
         handle( 0 );   //120 以下にする
         motor_f( 0, 0 );
         motor_r( 0, 0 );
        break;

  case 6:
         gain = 10;
         servoPwmOut( 0 );
              motor_f( 0, 0 );
         motor_r( 0, 0 );
              led_out( 0xf0 );
              causeout = lEncoderTotal;
     if(cnt1 >= 200){
            M_FreeMoter = 0;
            cnt1 = 0;
            pattern = 0;
            break;
        }
break;
```

```
case 11:
    /* 通常トレース */
            gain = 10;
    i = getServoAngle(); // SERVO_STEP;
        led_out(0xff);
            servoPwmOut( iServoPwm );
    if( i >= 130 ) {//170
        motor_f( 0, 0 );
        motor_r( 0, 0 );
   }
            else if( i >= 60 ) {
                if (iEncoder <=   TESTSPEED){   // 35 50
        motor_f( diff(80), 80 );   //50
        motor_r( diff(80), 80 );   //50
                }
                else{
                    motor_f( 0, 0 );
        motor_r( 0, 0 );
                }
    }
                    else if( i >= 40 ) {
                if (iEncoder <= TESTSPEED){   //40   55
        motor_f( diff(90), 90 );   //50
        motor_r( diff(90), 90 );   //50
                }
                else{
                    motor_f( 0, 0 );
        motor_r( 0, 0 );
                }
    }
            else if( i >= 20 ) {
                if (iEncoder <=   TESTSPEED){   //45 60
        motor_f( diff(100), 100 );
        motor_r( diff(100), 100 );
                }
                else{
                    motor_f( 0, 0 );
        motor_r( 0, 0 );
                }
    }
       else if( i <= -130 ) {//-170
        motor_f( 0, 0 );
        motor_r( 0, 0 );
    }
            else if( i <= -60 ) {
                if (iEncoder <=   50){//35
```

```
        motor_f( 80, diff(80) );
        motor_r( 80, diff(80) );
                }
                else{
                        motor_f( 0, 0 );
        motor_r( 0, 0 );
                }
}
else if( i <= -40 ) {
                if (iEncoder <=    TESTSPEED){ //40 55
        motor_f( 90, diff(90) );
        motor_r( 90, diff(90) );
                }
                else{
                        motor_f( 0, 0 );
        motor_r( 0, 0 );
                }
}
  else if( i <= -20 ) {
              if(iEncoder <=    TESTSPEED){ //45 60
        motor_f( 100, diff(100) );
        motor_r( 100, diff(100) );
}
else{
                        motor_f( 0, 0 );
        motor_r( 0, 0 );
            }
  }
            else {
                        if (iEncoder <=    TESTSPEED){ //50 70
        motor_f( 100, 100 );
        motor_r( 100, 100 );
                        }
                        else{
                  motor_f( 0, 0 );
        motor_r( 0, 0 );
                        }
            }

            if(center_inp() == 0x00){
                        led_out(0x00);
            }
            else {
                        led_out(0xff);
            }
```

```
if( check_crossline()) {          /* クロスラインチェック              */
                servoPwmOut( 0 );
                    lEncoderCrank = lEncoderTotal;
                    causeout = lEncoderTotal;
                    led_out(0x0f);
    pattern = 21;
                    break;
}

        if( check_zlineR()) {                                   // 右車線変更ラインチェック
                servoPwmOut( 0 );
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
                Lane_Change = RIGHT;                    // 1:左  0:右
                        led_out( 0x33 );
                pattern = 51;
                        break;
                    }

    if( check_zlineL()) {                                   // 左車線変更ラインチェック
                servoPwmOut( 0 );
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
                Lane_Change = LEFT;                             // 1:左  0:右
                        led_out( 0x44 );
                pattern = 51;
                        break;
                    }

        if( !check_Noline() ){
        causeout = lEncoderTotal;
        }


        if( pushsw_get() && cnt1 >= 1000) {
                M_FreeMoter = 0;
    setBeepPatternS( 0xcc00 );
                cnt1 = 0;
    pattern = 6;
    break;
}

/*          if (lEncoderTotal > (3100*200)){//2654?で 1m    692    3000
            pattern = 2;
            break;
            }*/
```

```
                break;


    case 21:
        /* クロスライン通過処理 */
            gain = 10;
                servoPwmOut( 0 );
                        led_out(0x00);
                        if (iEncoder <=   TESTSPEED){   //45 50
                                motor_f( 70, 70 );   //60
                            motor_r( 70, 70 );   //60
                        }
                        else if (iEncoder >   TESTSPEED){
                                motor_f( 0, 0);
                            motor_r( 0, 0);
                        }
                if((sensor_inp() != 0x0f) && (lEncoderTotal - lEncoderCrank) >= 30) { // 10mm で 3 300mm=90
center_inp() == 0x01
                            lEncoderCrank = lEncoderTotal;
                            causeout = lEncoderTotal;
                    pattern = 22;
                                break;
                        }
                 if( !check_Noline() ){
                causeout = lEncoderTotal;
                }
        break;


    case 22:
        /* クロスライン後のトレース、直角検出処理 */
                gain = 10;
                servoPwmOut( iServoPwm );
                        led_out(0x03);
                        if (iEncoder <= TESTSPEED){   //45   50
                                motor_f( 70, 70 );   //70
                            motor_r( 70, 70 );   //70
                        }
                        else if (iEncoder > TESTSPEED){
                                motor_f( 0, 0);
                            motor_r( 0, 0);
                        }
                if ( (lEncoderTotal - lEncoderCrank) >= 70 ) {
                                //crank_mode = 1;
                                if( sensor_inp()==0x01 || sensor_inp() == 0x03) {              // 右クランク
                                    crank_mode = 1;
                                lEncoderCrank = lEncoderTotal;
```

```
                                causeout = lEncoderTotal;
            led_out(0x1);
                    pattern = 31;
                                break;
                }
                if( sensor_inp() == 0x08 || sensor_inp() == 0x0c) {      // 左クランク
                            crank_mode = 1;
                lEncoderCrank = lEncoderTotal;
                                causeout = lEncoderTotal;
                                led_out(0x2);
                pattern = 41;
                                break;
                }
                }
        if( !check_Noline() ){
        causeout = lEncoderTotal;
        }
    break;

case 31:    // (右)クランク処理
                gain = 10;
                servoPwmOut( 0 );
                led_out(0x10);
                    if (iEncoder <= TESTSPEED){//43 50
                    motor_f( 70 , 70 );
                    motor_r( 70 , 70 );
            }
            else{
                    motor_f( 0, 0);
                    motor_r( 0, 0);
                }

        if(center_inp() == 0x00) { // 10mm で 3
                            lEncoderCrank = lEncoderTotal;
                            causeout = lEncoderTotal;
            pattern = 32;
                            break;
                }
        if( !check_Noline() ){
        causeout = lEncoderTotal;
        }
        break;

    case 32:    // (右)少し時間が経つまで待つ
        handle( 115 ); //100
                    led_out(0x20);
```

```
                if (iEncoder <= TESTSPEED){   //43 50
                        motor_f( 100, -50 );
                motor_r( 100, -50 );
                }
                else{
                        motor_f( 0, 0 );
                    motor_r( 0, 0 );
                }
                if(sensor_inp() != 0x04 || sensor_inp() != 0x08 || sensor_inp() != 0x0c){
        if((sensor_inp()==0x01) && (lEncoderTotal - lEncoderCrank) >= 60)   { // 10mm で 3
            iSensorPattern = 0;
                        crank_mode = 0;
            lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
            pattern = 33;
        }
        }
        break;

        case 33:
                handle(95); //90
                led_out(0x30);
                if (iEncoder <= TESTSPEED)      {   //43 50
                        motor_f( 100, 60 );   //50 30
                    motor_r( 100, 60 );   //50 30
                }
                else{
                        motor_f( 0, 0);
                    motor_r( 0, 0);
                }
                //左が反応していないか
                if(sensor_inp() != 0x04 || sensor_inp() != 0x08 || sensor_inp() != 0x0c){
                if(center_inp() == 0x01   && (lEncoderTotal - lEncoderCrank) >= 60){      //真ん中が反応して
いる
                lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
                        pattern = 34;
                 }
                }
                break;

        case 34:
            gain = 10;
        servoPwmOut( iServoPwm );
                led_out(0x40);
                if (iEncoder <=   TESTSPEED){ //43 50
```

```
                        motor_f( 70, 70 ); //60
                    motor_r( 70, 70 ); //60
                }
                else{
                        motor_f( 0, 0);
                    motor_r( 0, 0);
                }
            if( (lEncoderTotal - lEncoderCrank) >= 10 ) {
                        led_out(3);
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
            pattern = 11;
            }
            if( !check_Noline() ){
            causeout = lEncoderTotal;
            }
            break;


case 41:    // (左)クランク処理
            gain = 10;
                servoPwmOut( 0 );
                if (iEncoder <= TESTSPEED){//43 50
                        motor_f( 70 , 70 );
                        motor_r( 70 , 70 );
                }
                else{
                        motor_f( 0, 0);
                        motor_r( 0, 0);
                    }
            if( center_inp() == 0x00){//(lEncoderTotal - lEncoderCrank) > 6 ) { // 10mm で 3
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
            pattern = 42;
                    }
            if( !check_Noline() ){
            causeout = lEncoderTotal;
            }
            break;

        case 42:    // (左)少し時間が経つまで待つ
            handle( -110 );
                    if (iEncoder <= TESTSPEED){ //43 50
                        motor_f( -50, 100 );
                    motor_r( -50, 100 );
                }
```

```
        else{
                motor_f( 0, 0);
            motor_r( 0, 0);
        }

        if((sensor_inp() != 0x01 || sensor_inp() != 0x02 || sensor_inp() != 0x03) ){    //右センサか真ん中のセンサが反応していない
    if((sensor_inp()==0x08) && (lEncoderTotal - lEncoderCrank) > 60 ) { //左センサが反応してかつ距離から前回の地点を引いた値が 21 より大きい  10mm で 3
        iSensorPattern = 0;
                crank_mode = 0;
                    lEncoderCrank = lEncoderTotal;
                    causeout = lEncoderTotal;
        pattern = 43;
                    break;
     }
            }
    break;

    case 43:
            handle(-100);
            if (iEncoder <= TESTSPEED){//43 50
                motor_f( 60 , 100 );
            motor_r( 60 , 100 );
            }
            else{
                motor_f( 0, 0);
            motor_r( 0, 0);
            }
            if(sensor_inp() != 0x01 || sensor_inp() != 0x02 || sensor_inp() != 0x03){
            if(center_inp()==0x01 && (lEncoderTotal - lEncoderCrank) > 60 ) { // 10mm で 3
                lEncoderCrank = lEncoderTotal;
                causeout = lEncoderTotal;
                pattern = 44;
                break;
             }
            }
            break;

    case 44:
            gain = 10;
    servoPwmOut( iServoPwm );
            if (iEncoder <= TESTSPEED){//43 50
                motor_f( 70, 70);
            motor_r( 70, 70);
            }
```

```
                else{
                        motor_f( 0, 0 );
                    motor_r( 0, 0 );
                }
        if( (lEncoderTotal - lEncoderCrank) >= 10 ) {
                        led_out(3);
            cnt1 = 0;
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
            pattern = 11;
                        break;
        }
        if( !check_Noline() ){
        causeout = lEncoderTotal;
        }
    break;

 case 51:    // クランクチェック
          gain = 10;
        servoPwmOut( 0 );
                if (iEncoder <= TESTSPEED){//43 50
                        motor_f( 60, 60 );
                    motor_r( 60, 60 );
                }
                else if (iEncoder > TESTSPEED){
                        motor_f( 0, 0);
                    motor_r( 0, 0);
                }
        if(((lEncoderTotal - lEncoderCrank) <= 300) && check_crossline()) {
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
            pattern = 21;
                        break;
        }
        else if((lEncoderTotal - lEncoderCrank) > 300)   { // 10mm で 3
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
            pattern = 52;
                        break;
        }
        if( !check_Noline() ){
        causeout = lEncoderTotal;
        }
        break;

case 52:    // ライン終了サーチ
```

```
gain = 10;
servoPwmOut( iServoPwm );
        if (iEncoder <= TESTSPEED){//45   50
                motor_f( 80, 80 );
            motor_r( 80, 80 );
        }
        else if (iEncoder > TESTSPEED){
                motor_f( 0, 0); //-100
            motor_r( 0, 0);
        }

 if( check_Noline() && Lane_Change == RIGHT ) { //  右車線変更?
                crank_mode = 1;
                lEncoderCrank = lEncoderTotal;
                causeout = lEncoderTotal;
                led_out( 1 );
      pattern = 61;
                break;
 } else if( check_Noline() && Lane_Change == LEFT ) { //  左車線変更？
                crank_mode = 1;
                lEncoderCrank = lEncoderTotal;
                causeout = lEncoderTotal;
                led_out( 2 );
      pattern = 71;
                break;
      }
            causeout = lEncoderTotal;
 break;

  case 61:    // (右)車線変更処理 1
    handle( 50 );
        if (iEncoder <= TESTSPEED){//43 50
                motor_f( 80, 70 );
            motor_r( 80, 70 );
        }
        else if (iEncoder > TESTSPEED){
                motor_f( 0, 0);
            motor_r( 0, 0);
                }

 if( (sensor_inp() == 0x01) && (sensor_inp() != 0x08) ){       // デジタル 1,2,4   アナログ L,R タイヤ側から見て
                iSensorPattern = 0;
      crank_mode = 0;
                lEncoderCrank = lEncoderTotal;
                causeout = lEncoderTotal;
```

```
                pattern = 62;
                        break;
                }
        break;

case 62:    // (右)車線変更処理 2
                handle( 1 );
                if (iEncoder <= TESTSPEED){//43 50
        motor_f( 80, 80 );
                motor_r( 80, 80 );
                }
                else if (iEncoder > TESTSPEED){
                        motor_f( 0, 0);
                        motor_r( 0, 0);
                }

        if( (sensor_inp() == 0x08 || sensor_inp() == 0x04) ){//&& (sensor_inp() != 0x01)) {
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
                pattern = 63;
                        break;
                }
                break;

        case 63:    // (右)車線変更処理 3
  handle( 25 );
        if (iEncoder <= TESTSPEED){//43 50
        motor_f( 70, 80 );
                motor_r( 70, 80 );
                }
                else if (iEncoder > TESTSPEED){
                        motor_f( 0, 0);
                        motor_r( 0, 0);
                        }

        if( sensor_inp() == 0x02 || sensor_inp() == 0x01 || center_inp() == 0x01 ) {
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
                pattern = 64;
                        break;
                }
                break;

    case 64:  // (右)車線変更処理 4
                gain = 10;
        servoPwmOut( iServoPwm );
```

```
                if (iEncoder <= TESTSPEED){//43    50
                        motor_f( 60, 60 );
                    motor_r( 60, 60 );
                }
                else if (iEncoder > TESTSPEED){
                        motor_f( 0, 0);
                    motor_r( 0, 0);
                }

        if( (lEncoderTotal - lEncoderCrank) >= 25 ) { // 10mm で 3    150mm 後に通常トレースへ遷移
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
            pattern = 11;
                        break;
        }
if( !check_Noline() ){
        causeout = lEncoderTotal;
        }
        break;

  case 71:     // (左)車線変更処理 1
        handle( -50 );
                        if (iEncoder <= TESTSPEED){//43 50
                                motor_f( 70, 80 );
                            motor_r( 70, 80 );
                        }
                        else if (iEncoder > TESTSPEED){
                                motor_f( 0, 0);
                            motor_r( 0, 0);
                        }

                if( (sensor_inp() == 0x08)     && (sensor_inp() != 0x01)){
                    iSensorPattern = 0;
                    crank_mode = 0;
                                lEncoderCrank = lEncoderTotal;
                                causeout = lEncoderTotal;
                    pattern = 72;
                                break;
                        }
        break;

    case 72:     // (左)車線変更処理 2
                handle( -1 );
                  if (iEncoder <= TESTSPEED){//43 50
        motor_f( 80, 80 );
            motor_r( 80, 80 );
```

```
                }
                else if (iEncoder > TESTSPEED){
                        motor_f( 0, 0);
                        motor_r( 0, 0);
                }

        if( (sensor_inp() == 0x01 || sensor_inp() == 0x02)){ // && (sensor_inp() != 0x08)) {
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
                pattern = 73;
                        break;
                        }
                break;

        case 73:    // (左)車線変更処理 3
handle( 25 );
                if (iEncoder <= TESTSPEED){//43 50
        motor_f( 80, 70 );
                motor_r( 80, 70 );
                }
                else if (iEncoder > TESTSPEED){
                        motor_f( 0, 0);
                        motor_r( 0, 0);
                }

        if( sensor_inp() == 0x04 || sensor_inp() == 0x08 || center_inp() == 0x01) {
                        lEncoderCrank = lEncoderTotal;
                        causeout = lEncoderTotal;
                pattern = 74;
                        break;
                }
                break;

case 74:    // (左)車線変更処理 4
                gain = 10;
        servoPwmOut( iServoPwm );
                if (iEncoder <= TESTSPEED){//43   50
                        motor_f( 60, 60 );
                    motor_r( 60, 60 );
                }
                else if (iEncoder > TESTSPEED){
                        motor_f( 0, 0);
                    motor_r( 0, 0);
                }

        if((lEncoderTotal - lEncoderCrank) >= 25 ) { // 10mm で 3   150mm 後に通常トレースへ遷移
```

```
                                lEncoderCrank = lEncoderTotal;
                                    causeout = lEncoderTotal;
                            pattern = 11;
                                    break;
                        }
                        if( !check_Noline() ){
                        causeout = lEncoderTotal;
                        }
                        break;
        default:
            break;
          }
            }
}

/************************************************************************/
/* R8C/38A スペシャルファンクションレジスタ(SFR)の初期化                */
/************************************************************************/
void init( void )
{
    int        i;

    /* クロックを XIN クロック(20MHz)に変更 */
    prc0   = 1;                         /* プロテクト解除                */
    cm13   = 1;                         /* P4_6,P4_7 を XIN-XOUT 端子にする*/
    cm05   = 0;                         /* XIN クロック発振              */
    for(i=0; i<50; i++ );               /* 安定するまで少し待つ(約 10ms) */
    ocd2   = 0;                         /* システムクロックを XIN にする  */
    prc0   = 0;                         /* プロテクト ON                 */

    /* ポートの入出力設定 */

    /*  PWM(予備)       左前 M_PMW       右前 M_PWM       ブザー
        センサ左端      センサ左中       センサ右中       センサ右端  */
    p0     = 0x00;
    prc2 = 1;                           /* PD0 のプロテクト解除          */
    pd0    = 0xf0;

    /*  センサ中心      ｽﾀｰﾄﾊﾞｰ          RxD0             TxD0
        DIPSW3          DIPSW2          DIPSW1           DIPSW0          */
    pur0 |= 0x04;                       /* P1_3～P1_0 のプルアップ ON    */
    p1    = 0x00;
    pd1 = 0x10;

    /*  右前 M_方向     ステア M_方向   ステア M_PWM    右後 M_PWM
        右後 M_方向     左後 M_PWM      左後 M_方向      左前 M_方向      */
```

```
p2  = 0x00;
pd2 = 0xff;

/*  none            none            none            none
    none            none            none            エンコーダ A 相   */
p3  = 0x00;
pd3 = 0xfe;

/*  XOUT            XIN             ボード上の LED    none
    none            VREF            none            none            */
p4  = 0x20;                         /* P4_5 の LED:初期は点灯          */
pd4 = 0xb8;

/*  none            none            none            none
    none            none            none            none            */
p5  = 0x00;
pd5 = 0xff;

/*  none            none            none            none
    none            none            none            none            */
p6  = 0x00;
pd6 = 0xff;

/*  CN6.2 入力        CN6.3 入力        CN6.4 入力        CN6.5 入力
    none(ｱﾅﾛｸﾞ 予備) 角度 VR          センサ_左ｱﾅﾛｸﾞ   センサ_右ｱﾅﾛｸﾞ   */
p7  = 0x00;
pd7 = 0x00;

/*  DIPSWorLED      DIPSWorLED      DIPSWorLED      DIPSWorLED
    DIPSWorLED      DIPSWorLED      DIPSWorLED      DIPSWorLED      */
pur2 |= 0x03;                       /* P8_7～P8_0 のプルアップ ON      */
p8  = 0x00;
pd8 = 0x00;

/*  -               -               ﾌﾟｯｼｭｽｲｯﾁ         P8 制御(LEDorSW)
    右前 M_Free     左前 M_Free     右後 M_Free     左後 M_Free     */
p9  = 0x00;
pd9 = 0x1f;
pu23 = 1;   // P9_4,P9_5 をプルアップする
/* タイマ RB の設定 */
/* 割り込み周期 = 1 / 20[MHz]   * (TRBPRE+1) * (TRBPR+1)
                = 1 / (20*10^6) * 200        * 100
                = 0.001[s] = 1[ms]
*/
trbmr  = 0x00;                      /* 動作モード、分周比設定          */
trbpre = 200-1;                     /* プリスケーラレジスタ            */
```

```
    trbpr   = 100-1;                    /*  プライマリレジスタ              */
    trbic   = 0x06;                     /*  割り込み優先レベル設定          */
    trbcr   = 0x01;                     /*  カウント開始                    */

    /* A/D コンバータの設定 */
    admod    = 0x33;                     /*  繰り返し掃引モードに設定        */
    adinsel = 0x90;                     /*  入力端子 P7 の 4 端子を選択     */
    adcon1   = 0x30;                    /* A/D 動作可能                     */
    asm(" nop ");                       /* φAD の 1 サイクルウエイト入れる*/
    adcon0   = 0x01;                    /* A/D 変換スタート                 */

    /*  タイマ RG  タイマモード(両エッジでカウント)の設定 */
    timsr = 0x40;                       /* TRGCLKA 端子  P3_0 に割り当てる */
    trgcr = 0x15;                       /* TRGCLKA 端子の両エッジでカウント*/
    trgmr = 0x80;                       /* TRG のカウント開始               */

    /*  タイマ RC PWM モード設定(左前モータ、右前モータ) */
    trcpsr0 = 0x40;                     /* TRCIOA,B 端子の設定              */
    trcpsr1 = 0x33;                     /* TRCIOC,D 端子の設定              */
    trcmr    = 0x0f;                    /* PWM モード選択ビット設定         */
    trccr1   = 0x8e;                    /*  ｿｰｽｶｳﾝﾄ:f1,初期出力の設定      */
    trccr2   = 0x00;                    /*  出力レベルの設定                */
    trcgra   = TRC_MOTOR_CYCLE - 1;         /*  周期設定                        */
    trcgrb   = trcgrb_buff = trcgra;    /* P0_5 端子の ON 幅(左前モータ)    */
    trcgrc   = trcgrc_buff = trcgra;
//  trcgrc   = trcgra;                  /* P0_7 端子の ON 幅(予備)          */
    trcgrd   = trcgra;
//  trcgrd   = trcgrd_buff = trcgra;    /* P0_6 端子の ON 幅(右前モータ)    */
    trcic    = 0x07;                    /*  割り込み優先レベル設定          */
    trcier   = 0x01;                    /* IMIA を許可                      */
    trcoer   = 0x01;                    /*  出力端子の選択                  */
    trcmr   |= 0x80;                    /* TRC カウント開始                 */

    /*  タイマ RD  リセット同期 PWM モード設定(左後ﾓｰﾀ、右後ﾓｰﾀ、ｻｰﾎﾞ ﾓｰﾀ) */
    trdpsr0 = 0x08;                     /* TRDIOB0,C0,D0 端子設定           */
    trdpsr1 = 0x05;                     /* TRDIOA1,B1,C1,D1 端子設定        */
    trdmr    = 0xf0;                    /*  バッファレジスタ設定            */
    trdfcr   = 0x01;                    /*  リセット同期 PWM モードに設定   */
    trdcr0   = 0x20;                    /*  ソースカウントの選択:f1         */
    trdgra0 = trdgrc0 = TRD_MOTOR_CYCLE - 1;    /*  周期設定                */
    trdgrb0 = trdgrd0 = 0;              /* P2_2 端子の ON 幅(左後モータ)    */
    trdgra1 = trdgrc1 = 0;              /* P2_4 端子の ON 幅(右後モータ)    */
    trdgrb1 = trdgrd1 = 0;              /* P2_5 端子の ON 幅(サーボモータ) */
    trdoer1 = 0xcd;                     /*  出力端子の選択                  */
    trdstr   = 0x0d;                    /* TRD0 カウント開始                */
}
```

```
/************************************************************************/
/*  タイマ RB 割り込み処理                                              */
/************************************************************************/
#pragma interrupt /B intTRB(vect=24)
void intTRB( void ){
    unsigned int i;

    asm(" fset I ");                    /*  タイマ RB 以上の割り込み許可    */

    cnt1++;

    /*  サーボモータ制御 */
    servoControl();
        handle();

    /*  ブザー処理 */
    beepProcessS();

    /* 10 回中 1 回実行する処理 */
        /*都合上 4 回にしている*/
    iTimer10++;
    switch( iTimer10 ) {
    case 1:
        /*  エンコーダ制御 */
        i = trg;
        iEncoder            = i - uEncoderBuff;
        lEncoderTotal += iEncoder;
                p4_5 = trg;
        uEncoderBuff = i;
        break;

    case 2:
        /*  スイッチ読み込み準備 */
        p9_4 = 0;                       /* LED 出力 OFF                 */
        pd8   = 0x00;
        break;

    case 3:
        /*  スイッチ読み込み、LED 出力 */
        types_dipsw = ~p8;              /* ﾄﾞﾗｲﾌﾞ基板 TypeS Ver.3 の SW 読み込み*/
        p8   = types_led;               /* ﾄﾞﾗｲﾌﾞ基板 TypeS Ver.3 の LED へ出力*/
        pd8 = 0xff;
        p9_4 = 1;                       /* LED 出力 ON                  */
        break;
```

```
    case 7: //前回は 4 だった
        /* iTimer10 変数の処理 */
        iTimer10 = 0;
        break;
    }
}

/************************************************************************/
/* タイマ RC 割り込み処理                                               */
/************************************************************************/
#pragma interrupt intTRC(vect=7)
void intTRC( void )
{
    trcsr &= 0xfe;

    /* タイマ RC　デューティ比の設定 */
    trcgrb = trcgrb_buff;
    trcgrd = trcgrd_buff;
}

/************************************************************************/
/* アナログセンサ基板 TypeS Ver.2 のデジタルセンサ値読み込み            */
/* 引数　 なし                                                          */
/* 戻り値 左端、左中、右中、右端のデジタルセンサ 0:黒 1:白              */
/************************************************************************/
unsigned char sensor_inp( void )　 //0,0,0,0　機体から見て
{
    unsigned char sensor;

    sensor = ~p0 & 0x0f;

    return sensor;
}

/************************************************************************/
/* アナログセンサ基板 TypeS Ver.2 の中心デジタルセンサ読み込み          */
/* 引数　 なし                                                          */
/* 戻り値 中心デジタルセンサ 0:黒 1:白                                  */
/************************************************************************/
unsigned char center_inp( void )
{
    unsigned char sensor;

    sensor = ~p1_7 & 0x01;

    return sensor;
```

```
}

/************************************************************************/
/* アナログセンサ基板 TypeS Ver.2 のスタートバー検出センサ読み込み          */
/* 引数    なし                                                         */
/* 戻り値 0:スタートバーなし 1:スタートバーあり                          */
/************************************************************************/
unsigned char startbar_get( void )
{
    unsigned char sensor;

    sensor = ~p1_6 & 0x01;

    return sensor;
}

/************************************************************************/
/* マイコンボード上のディップスイッチ値読み込み                          */
/* 引数    なし                                                         */
/* 戻り値 スイッチ値 0～15　分けている                                   */
/************************************************************************/
unsigned char dipsw_get( void )
{
    unsigned char sw;

    sw = p1 & 0x0e;                     /* P1_3～P1_1 読み込み          */

    return sw;
}

unsigned char dipsw_getf( void )
{
    unsigned char sw;

    sw = p1 & 0x01;                     /*P1_0 読み込み                 */

    return sw;
}
/************************************************************************/
/* モータドライブ基板 TypeS Ver.3 上のディップスイッチ値読み込み         */
/* 引数    なし                                                         */
/* 戻り値 スイッチ値 0～255　意味なし                                    */
/************************************************************************/
unsigned char dipsw_get2( void )
{
    /* 実際の入力はタイマ RB 割り込み処理で実施 */
```

```
        unsigned char sw;

    sw = p8 & 0xf0;                     /*P8_7～P8_4 読み込み              */

    return sw;
  //  return types_dipsw;
}


unsigned char dipsw_get3( void )
{
    /* 実際の入力はタイマ RB 割り込み処理で実施 */
        unsigned char sw;

    sw = p8 & 0x03;                     /* P8_1～P8_0 読み込み              */

    return sw;
 //   return types_dipsw;
}
unsigned char dipsw_get4( void )
{
    /* 実際の入力はタイマ RB 割り込み処理で実施 */
        unsigned char sw;

    sw = p8 & 0x0c;                     /* P8_3～P8_2 読み込み              */

    return sw;
 //   return types_dipsw;
}

/************************************************************************/
/* モータドライブ基板 TypeS Ver.3 上のプッシュスイッチ値読み込み          */
/* 引数    なし                                                         */
/* 戻り値 スイッチ値 0:OFF 1:ON                                          */
/************************************************************************/
unsigned char pushsw_get( void )
{
    unsigned char sw;

    sw = ~p9_5 & 0x01;

    return sw;
}

/************************************************************************/
/* モータドライブ基板 TypeS Ver.3 の CN6 の状態読み込み                  */
```

```
/* 引数     なし                                                              */
/* 戻り値  0～15                                                              */
/****************************************************************************/
unsigned char cn6_get( void )
{
    unsigned char data;

    data = p7 >> 4;

    return data;
}

/****************************************************************************/
/* モータドライブ基板 TypeS Ver.3 の LED 制御                                  */
/* 引数     8 個の LED 制御  0:OFF 1:ON                                       */
/* 戻り値  なし                                                               */
/****************************************************************************/
void led_out( unsigned char led )
{
    /* 実際の出力はタイマ RB 割り込み処理で実施 */
    types_led = led;
}

/****************************************************************************/
/* 後輪の速度制御                                                             */
/* 引数     左モータ:-100～100 , 右モータ:-100～100                           */
/*          0 で停止、100 で正転 100%、-100 で逆転 100%                       */
/* 戻り値  なし                                                               */
/****************************************************************************/
void motor2_r( int accele_l, int accele_r ){

                int sw_data;

                if( M_FreeMoter == 1 ) {
        accele_l = 1;
                accele_r = 1;
    }

    sw_data   = dipsw2_pattern[dipsw_get2()] + 5;          /* ディップスイッチ読み込み      */
    accele_l = -accele_l * sw_data / 20;
    accele_r = -accele_r * sw_data / 20;


    // 左後モータ
    if( accele_l >= 0 ) {
        p2_1 = 0;
```

```
        trdgrd0 = (long)( TRD_MOTOR_CYCLE - 2 ) * accele_l / 100;
    } else {
        p2_1 = 1;
        trdgrd0 = (long)( TRD_MOTOR_CYCLE - 2 ) * ( -accele_l ) / 100;
    }

    // 右後モータ
    if( accele_r >= 0 ) {
        p2_3 = 0;
        trdgrc1 = (long)( TRD_MOTOR_CYCLE - 2 ) * accele_r / 100;
    } else {
        p2_3 = 1;
        trdgrc1 = (long)( TRD_MOTOR_CYCLE - 2 ) * ( -accele_r ) / 100;
    }
}




/*********************************************************************/
/* 後輪の速度制御 2 ディップスイッチには関係しない motor 関数                    */
/* 引数    左モータ:-100～100 , 右モータ:-100～100                          */
/*          0 で停止、100 で正転 100%、-100 で逆転 100%                        */
/* 戻り値 なし                                                          */
/*********************************************************************/
void motor_r( int accele_l2, int accele_r2 )
{
     int accele_l,accele_r;

        if( M_FreeMoter == 1 ) {
                acceleFree = 1;
        accele_l = acceleFree;
                accele_r = acceleFree;
    }
    else {
                accele_l = accele_l2;
                accele_r = accele_r2;
        }

    // 左後モータ
    if( accele_l >= 0 ) {
        p2_1 = 1;
        trdgrd0 = (long)( TRD_MOTOR_CYCLE - 2 ) * accele_l / 100;
    } else {
        p2_1 = 0;
```

```
        trdgrd0 = (long)( TRD_MOTOR_CYCLE - 2 ) * ( -accele_l ) / 100;
    }

    // 右後モータ
    if( accele_r >= 0 ) {
        p2_3 = 1;
        trdgrc1 = (long)( TRD_MOTOR_CYCLE - 2 ) * accele_r / 100;
    } else {
        p2_3 = 0;
        trdgrc1 = (long)( TRD_MOTOR_CYCLE - 2 ) * ( -accele_r ) / 100;
    }
}



/*****************************************************************************/
/* 前輪の速度制御                                                            */
/* 引数    左モータ:-100～100 , 右モータ:-100～100                           */
/*           0 で停止、100 で正転 100%、-100 で逆転 100%                     */
/* 戻り値 なし                                                               */
/*****************************************************************************/
void motor2_f( int accele_l, int accele_r ){
    int sw_data;

        if( M_FreeMoter == 1 ) {
        accele_l = 1;
                accele_r = 1;
    }

    sw_data   = dipsw2_pattern[dipsw_get2()] + 5;          /* ディップスイッチ読み込み      */
    accele_l = accele_l * sw_data / 20;
    accele_r = accele_r * sw_data / 20;


    // 左前モータ
    if( accele_l >= 0 ) {
        p2_0 = 0;
    } else {
        p2_0 = 1;
        accele_l = -accele_l;
    }
    if( accele_l <= 5 ) {
        trcgrb = trcgrb_buff = trcgra;
    } else {
        trcgrb_buff = (unsigned long)(TRC_MOTOR_CYCLE-2) * accele_l / 100;
    }
```

```
    // 右前モータ
    if( accele_r >= 0 ) {
        p2_7 = 0;
    } else {
        p2_7 = 1;
        accele_r = -accele_r;
    }
    if( accele_r <= 5 ) {
        trcgrd = trcgrd_buff = trcgra;
    } else {
        trcgrd_buff = (unsigned long)(TRC_MOTOR_CYCLE-2) * accele_r / 100;
    }
}




/*****************************************************************************/
/* 前輪の速度制御 2 ディップスイッチには関係しない motor 関数                    */
/* 引数    左モータ:-100～100 , 右モータ:-100～100                              */
/*          0 で停止、100 で正転 100%、-100 で逆転 100%                          */
/* 戻り値 なし                                                                  */
/*****************************************************************************/
void motor_f( int accele_l2, int accele_r2 )
{
        int accele_l,accele_r;

        if( M_FreeMoter == 1 ) {
                acceleFree = 1;
        accele_l = acceleFree;
                accele_r = acceleFree;
    }
    else {
                accele_l = accele_l2;
                accele_r = accele_r2;
        }

    /* 左前モータ */
    if( accele_l >= 0 ) {
        p2_0 = 0;
    } else {
        p2_0 = 1;
        accele_l = -accele_l;
    }
    if( accele_l <= 5 ) {
```

```
        trcgrb = trcgrb_buff = trcgra;
    } else {
        trcgrb_buff = (unsigned long)(TRC_MOTOR_CYCLE-2) * accele_l / 100;
    }

    /* 右前モータ */
    if( accele_r >= 0 ) {
        p2_7 = 0;
    } else {
        p2_7 = 1;
        accele_r = -accele_r;
    }
    if( accele_r <= 5 ) {
        trcgrd = trcgrd_buff = trcgra;
    } else {
        trcgrd_buff = (unsigned long)(TRC_MOTOR_CYCLE-2) * accele_r / 100;
    }
}

/*****************************************************************/
/* 後モータ停止動作（ブレーキ、フリー）                          */
/* 引数    左モータ:FREE or BRAKE , 右モータ:FREE or BRAKE       */
/* 戻り値 なし                                                   */
/*****************************************************************/
void motor_mode_r( int mode_l, int mode_r )
{
    if( mode_l ) {
        p9_0 = 1;
    } else {
        p9_0 = 0;
    }
    if( mode_r ) {
        p9_1 = 1;
    } else {
        p9_1 = 0;
    }
}

/*****************************************************************/
/* 前モータ停止動作（ブレーキ、フリー）                          */
/* 引数    左モータ:FREE or BRAKE , 右モータ:FREE or BRAKE       */
/* 戻り値 なし                                                   */
/*****************************************************************/
void motor_mode_f( int mode_l, int mode_r )
{
    if( mode_l ) {
```

```
        p9_2 = 1;
    } else {
        p9_2 = 0;
    }
    if( mode_r ) {
        p9_3 = 1;
    } else {
        p9_3 = 0;
    }
}

/**********************************************************************/
/* サーボモータ制御                                                   */
/* 引数    サーボモータ PWM：-100～100                                */
/*           0 で停止、100 で正転 100%、-100 で逆転 100%              */
/* 戻り値 なし                                                        */
/**********************************************************************/
void servoPwmOut( int pwm )
{
    if( pwm >= 0 ) {
        p2_6 = 0;
        trdgrd1 = (long)( TRD_MOTOR_CYCLE - 2 ) * pwm / 100;
    } else {
        p2_6 = 1;
        trdgrd1 = (long)( TRD_MOTOR_CYCLE- 2 ) * ( -pwm ) / 100;
    }
}

/**********************************************************************/
/* クロスライン検出処理                                               */
/* 引数    なし                                                       */
/* 戻り値 0:クロスラインなし 1:あり                                   */
/**********************************************************************/
int check_crossline( void )
{
    int ret;
        ret = 0;

   //   if((sensor_inp()==0x0f   ||   sensor_inp()==0x0e   ||   sensor_inp()==0x0d   ||   sensor_inp()==0x0b   ||
sensor_inp()==0x07)) {
    //sensor_inp()!=0x03 || sensor_inp()!=0x0c || sensor_inp()!=0x01 || sensor_inp()!=0x08)
          if((sensor_inp()==0x09 || sensor_inp()==0x05 || sensor_inp()==0x0a || sensor_inp()==0x0f
            || sensor_inp()==0x0e || sensor_inp()==0x0d || sensor_inp()==0x0b || sensor_inp()==0x07)
            && center_inp() == 0x01){
        ret = 1;
    }
```

```
    return ret;
}
/*********************************************************************/
/* 右車線変更ライン検出処理                                          */
/* 引数    なし                                                      */
/* 戻り値 0:車線変更ラインなし 1:あり                                */
/*********************************************************************/
int check_zlineR( void )
{
    int ret;
    ret = 0;

  //   if((sensor_inp()==0x03) || (sensor_inp()==0x01)) {
    //sensor_inp()!=0x09 || sensor_inp()!=0x05 || sensor_inp()!=0x0a || sensor_inp()!=0x0f)
            if((sensor_inp()==0x01 || sensor_inp()==0x03)
              && center_inp() == 0x01){
              ret = 1;
    }
    return ret;
}

/*********************************************************************/
/* 左車線変更ライン検出処理                                          */
/* 引数    なし                                                      */
/* 戻り値 0:車線変更ラインなし 1:あり                                */
/*********************************************************************/
int check_zlineL( void )
{
    int ret;
    ret = 0;

  //   if((sensor_inp()==0x0c) || (sensor_inp()==0x08)) {
    ///sensor_inp()!=0x09 || sensor_inp()!=0x05 || sensor_inp()!=0x0a || sensor_inp()!=0x0f)
            if((sensor_inp()==0x08 || sensor_inp()==0x0c)
                  && center_inp() == 0x01){
              ret = 1;
    }
    return ret;
}

/*********************************************************************/
/* ライン終了検出処理                                                */
/* 引数    なし                                                      */
/* 戻り値 0:車線変更ラインなし 1:あり                                */
/*********************************************************************/
int check_Noline( void )
```

```
{
    int ret;
    ret = 0;

    if( (sensor_inp()==0x00) && center_inp()==0x00) {
        ret = 1;
    }
    return ret;
}



/*********************************************************************/
/* サーボ角度取得                                                    */
/* 引数    なし                                                      */
/* 戻り値 入れ替え後の値                                             */
/*********************************************************************/
int getServoAngle( void )
{
    return( ad2 - iAngle0 );
}

/*********************************************************************/
/* アナログセンサ値取得                                              */
/* 引数    なし                                                      */
/* 戻り値 センサ値                                                   */
/*********************************************************************/
int getAnalogSensor( void )
{
    int ret;

    //ret = ad1 - ad0;                    /* アナログセンサ情報取得       */

    if( !crank_mode ) {
       //   クランクモードでなければ補正処理
        switch( iSensorPattern ) {
        case 0:
            if( sensor_inp() == 0x04 && center_inp() == 0x00) {
                ret = -650;    //-650
                break;
            }
            else if( sensor_inp() == 0x02 && center_inp() == 0x00) {
                ret = 650;     //650
                break;
            }
            else if( sensor_inp() == 0x08 && center_inp() == 0x00) {
                ret = -700;    //-700
```

```
                iSensorPattern = 1;
                break;
            }
            else if( sensor_inp() == 0x01 && center_inp() == 0x00) {
                ret = 700;      //700
                iSensorPattern = 2;
                break;
            }
                else {
                        ret = (ad1 - 10) - (ad0 + 10);
                }
            break;

        case 1:
            // センサ右寄り
            ret = -700;
            if(center_inp() == 0x01 || sensor_inp() == 0x04) {
                iSensorPattern = 0;
            }
            break;

        case 2:
            // センサ左寄り
            ret = 700;
            if(center_inp() == 0x01 || sensor_inp() == 0x02) {
                iSensorPattern = 0;
            }
            break;
        }
    }
        else{
                        ret = (ad1 - 10) - (ad0 + 10);
                }

    return ret;
}

/*********************************************************************/
/* サーボモータ制御                                                  */
/* 引数    なし                                                      */
/* 戻り値 グローバル変数 iServoPwm に代入                            */
/*********************************************************************/
void servoControl( void )
{
    int i, iRet, iP, iD;
    int kp, kd;
```

```
    i = getAnalogSensor();                          /* センサ値取得                    */
    kp =   dipsw1_pattern[dipsw_get()] ;            /* 調整できたら P,D 値は固定値に   */
    kd =   dipsw1_pattern[dipsw_get()]   * 10;      /* してください                    */


    /* サーボモータ用 PWM 値計算 */
     iP = gain * i;                                 /* 比例                            */
        //iP = kp * i;                              /* 比例                            */
        //10 を gain にする
    //iD = 100 * (iSensorBefore - i );              /* 微分(目安は P の 5〜10 倍)       */
        //iD = kd * (iSensorBefore - i );           /* 微分(目安は P の 5〜10 倍)       */
        iD = (gain * 10) * (iSensorBefore - i);     /* 微分(目安は P の 5〜10 倍)       */
    iRet = iP - iD;
    iRet = iRet / (130/2);   //iRet /= 64;         //64

    /* PWM の上限の設定 */
    if( iRet >   120 ) iRet =   120;                /* マイコンカーが安定したら        */
    if( iRet < -120 ) iRet = -120;                  /* 上限を 90 くらいにしてください */


        iServoPwm = iRet;

    iSensorBefore = i;                              /* 次回はこの値が 1ms 前の値となる*/
}


/************************************************************************/
/* モジュール名 handle */
/* 処理概要 サーボモータ制御 角度指定用 */
/* 引数 なし */
/* 戻り値 グローバル変数 servoPwmOut に代入 */
/************************************************************************/
void handle( int iSetAngle ){
    int i, j, iRet, iP, iD ;
    i = - iSetAngle; /* 設定したい角度 */
    //j = ( p7_2 >>2 );
    j = getServoAngle(); /* 現在の角度 */
    /* サーボモータ用 PWM 値計算 */
    iP = 10 * (j - i); /* 比例 */
    iD = 100 * (iAngleBefore2 - j); /* 微分 */
    iRet = iP - iD;
    iRet = iRet / (7/2);   //iRet /= 4;   //2
    if( iRet >   120 ) iRet =   120; /* マイコンカーが安定したら */
    if( iRet < -120 ) iRet = -120; /* 上限を 90 くらいにしてください */
```

```
    servoPwmOut(iRet);
    iAngleBefore2 = j;
}



/************************************************************************/
/* 外輪の PWM から、内輪の PWM を割り出す　ハンドル角度は現在の値を使用　　*/
/* 引数　　外輪 PWM                                                      */
/* 戻り値　内輪 PWM                                                      */
/************************************************************************/
int diff( int pwm )
{
    int i, ret;

    i = getServoAngle() / SERVO_STEP;       /* 1 度あたりの増分で割る          */
    if( i <   0 ) i = -i;
    if( i > 40 ) i = 40;   //45
    ret = revolution_difference[i] * pwm / 100;

    return ret;
}



/************************************************************************/
/* end of file                                                          */
/************************************************************************/
```