```
/*********************************************************************
    GOTO MARK-X Motor Drive Universul Controler

    Compiler  : HI-TECH C
    CPU       : PIC16F628A(PIC16F648Aでも可)
    Oscillator: external 4MHz
    1-2相励磁でのクロック計算を基本とする

    2009/9/6～
    Copyright(C) 2009,2011 by embeddedark
 *********************************************************************/
#include    <htc.h>

//__CONFIG(XT & WDTDIS & PWRTEN & BOREN & MCLREN & LVPDIS & UNPROTECT);
//__CONFIG(XT & WDTDIS & PWRTEN & BORDIS & MCLREN & LVPDIS & UNPROTECT);
//__CONFIG(XT & WDTDIS & PWRTEN & BORDIS & MCLRDIS & LVPDIS & UNPROTECT);
__CONFIG(INTCLK & WDTDIS & PWRTEN & BORDIS & MCLRDIS & LVPDIS & UNPROTECT);

//__delay_us(), __delay_ms()の制御
#ifndef _XTAL_FREQ
    #define _XTAL_FREQ 4000000 //PICのクロックをHzで設定
#endif

/*=======================================
    BUILD OPTION
  =======================================*/
#undef  SLOWCTL
#undef  EEPTEST
#define ROTARY_COMP
#define OPTION_CHANGE
#undef MODE_INPUT_TEST
#undef INPUT_TEST

/*------------------------------
    I/O Difine
  ------------------------------*/
#define HIGH_LEVEL  (1)
#define LOW_LEVEL   (0)
#define SELECT      LOW_LEVEL
#define NOSELECT    HIGH_LEVEL
#define ON          (1)
#define OFF         (0)
#define OUTB_       RA0             // PIN17 MORTOR OUTPUT PORT B_
#define OUTA_       RA1             // PIN18 MORTOR OUTPUT PORT A_
#define OUTB        RA2             // PIN 1 MORTOR OUTPUT PORT B
#define OUTA        RA3             // PIN 2 MORTOR OUTPUT PORT A
#define IS_OUTB_(x) ((x)&0x01)
#define IS_OUTA_(x) ((x)&0x02)
#define IS_OUTB(x)  ((x)&0x04)
#define IS_OUTA(x)  ((x)&0x08)
//#define IN_MOTOR_TYPE RA5         // PIN 4 MCLR_ MORTOR TYPE(external pull up)  1: GOTO STD / 0: direct
#define IN_SPEED    RA4             // HIGH SPEED (external pull up)
#define IN_STOP     RA5             // STOP (external pull up)
#define IN_OSC2     RA6             // PIN15 OSC2
#define IN_OSC1     RA7             // PIN16 OSC1
#define IN_DIR      RB0             // REVERSE

#define SET_TRISA   (0b11110000)    // RA0(LSB)～RA3:OUTPUT,RA4～RA7(MSB):INPUT
#define INIT_PORTA  (0b00000000)    // PORTA INITIAL OUTPUT DATA
#define SET_TRISB   (0b11111011)    // OUTPUT TX(RB2)

#define SET_CMCON   (0b00000111)    // PORTA comparator off
#define UART_RX     RB1             // PIN 7 UART RX
#define UART_TX     RB2             // PIN 8 UART TX

/*=============================================
    TIMER COUNTER
  =============================================*/
#define HSPEED      3           // 恒星時比増速時スピード倍数設定
#define STD_PULSE   (7459200)   //((126 * 600 * 48 * 74) / 36)
#define DIRECT_PULSE    (3628800)   //(126 * 600 * 48)
#define STAR_RATE   (86164.091)
#define KINGS_RATE  (86190)     // 天文ガイドインタラクティブの記事の値は86200
                                // でもインターネットサイトでは86190が採用されている例が多い
#define SUN_RATE    (86400)
#define MOON_RATE   (89428.3)
#define STAR_HALF_RATE  (STAR_RATE*2)
#define RATCNT(x,y) (unsigned short)((((float)(_XTAL_FREQ/4) * (float)(x)) / (float)(y)))
#define HDATA(x)    ((x)>>8)
#define LDATA(x)    (x)

#define STAR_RATE_STD_CNT   RATCNT(STAR_RATE,STD_PULSE)             // 11551/23745
#define KINGS_RATE_STD_CNT  RATCNT(KINGS_RATE,STD_PULSE)            // 11556/23754
#define SUN_RATE_STD_CNT    RATCNT(SUN_RATE,STD_PULSE)              // 11583/23809
#define MOON_RATE_STD_CNT   RATCNT(MOON_RATE,STD_PULSE)             // 11989/24644
#define STAR_HALF_RATE_STD_CNT  RATCNT(STAR_HALF_RATE,STD_PULSE)    // 23102/47489

/*====================================
    EEPROM DATA
  ====================================*/
#define INIT_REIJI_MODE 1       /* 0:1-2相励磁/1:2-2相励磁*/
#define INIT_MOTOR_TYPE 0       /* 0:GOTO STANDARD/1:GOTO DIRECT */
#define INIT_LOCK       1       /* 0:UNLOCK/1:LOCK */
/* REIJI_MODE(0:1-2/1:2-2),MOTOR_TYPE(0:GOTO STD/1:DIRECT),LOCK_TYPE,NC,CUSTOM_TIMER_H,CUSTOM_TIMER_L,NC,NC */
__EEPROM_DATA(INIT_REIJI_MODE,INIT_MOTOR_TYPE,INIT_LOCK,0,0,0,0,0);

/* EEPROM DATA INDEX */
#define EEP_REIJI_IDX       0
#define EEP_MOTORTYPE_IDX   1
#define EEP_LOCK_IDX        2
#define EEP_CUSTOM_TIMER_H  4
#define EEP_CUSTOM_TIMER_L  5

#ifdef EEPTEST
#define EEP_SW_IDX      8
#define EEP_MODESW_IDX  9
#define EEP_TBL_LAST    9
#else
#undef  EEP_SW_IDX
#undef  EEP_MODESW_IDX
#define EEP_TBL_LAST    7
#endif

#define EEP_TBL_MAX     (EEP_TBL_LAST+1)

#if 0
/*====== GOTO MARK-X STANDARD MORTOR ======*/
/* STAR=11551,KINGS=11556,SUN=11583,MOON=11989*/
__EEPROM_DATA(0x2d,0x1f,0x2d,0x24,0x2d,0x3f,0x2e,0xd5);
/* HALF STAR=23102,DEC=3xSTAR=3850*/
__EEPROM_DATA(0x5a,0x3e,0x0f,0x0a,0,0,0,0);

/*====== GOTO MARK-X DIRECT MORTOR ======*/
/* STAR=23745,KINGS=23754,SUN=23809,MOON=24644*/
__EEPROM_DATA(0x5c,0xc1,0x5c,0xca,0x5d,0x01,0x60,0x44);
```

```
/* HALF_STAR=47489,DEC=3xSTAR=7915*/
__EEPROM_DATA(0xb9,0x81,0x1e,0xeb,0,0,0,0);

#endif
/*--------------------------------
    EEPROM DATA AREA
  --------------------------------*/
static unsigned char eepdata[EEP_TBL_MAX];


/*----------------------
    softwer sw asignment
  ----------------------*/
#define STOP_BIT    3
#define SPEED_BIT   4
#define DIR_BIT     5
#define STOP_MASK   (0x01<<STOP_BIT)    // readsw() return bit mask
#define SPEED_MASK  (0x01<<SPEED_BIT)   // readsw() return bit mask
#define DIR_MASK    (0x01<<DIR_BIT)     // readsw() return bit mask
#define IN_STOP_SW(x)   (((x)&STOP_MASK)>>STOP_BIT)
#define IN_SPEED_SW(x)  (((x)&SPEED_MASK)>>SPEED_BIT)
#define IN_DIR_SW(x)    (((x)&DIR_MASK)>>DIR_BIT)

#define IN_MODE_SW(x)   ((x)&0x07)  // ROTARY SW LOWER 3bit : readsw() return bit mask

#define SET_T1CON   (0b00000000)    // T1CKPS=0,T1OSCEN=0,T1SYNC=0,TMR1CS=0,TMR1ON=0
#define INIT_PORTB  (0b00000001)    // KEY STROBE OFF and first phase pulse

/* operation mode */
#define MODE_STAR_RATE  (0)
#define MODE_KINGS_RATE (1)
#define MODE_SUN_RATE   (2)
#define MODE_MOON_RATE  (3)
#define MODE_HALF_STAR_RATE (4)
#define MODE_DEC    (5)
#define MODE_CUSTOM (6)
#define MODE_REMOTE (7)

#define MOTOR_MARKX     (0)             // MOTORTYPE = GOTO STD MARK-X
#define MOTOR_DIRECT    (1)             // MOTORTYPE = DIRECT 1/600 MOTOR for GOTO MARK-X
#define CHATTER_DELAY   (5)

static unsigned char motor_type = INIT_MOTOR_TYPE;
static unsigned char reiji_mode = INIT_REIJI_MODE;
static unsigned char eeplock = 0;       // 0 : EEPROM WRITE UNLOCK / 1 : EEPROM WRITE LOCK
static unsigned char reversesw = 0;     // 1 : REVERSE  / 0 : FORWORD
static unsigned char reverse = 0;
static unsigned char highspdsw = 0;     // 0 : NORMAL   / 1 : HIGH SPEED
static unsigned char highspd = 0;
static unsigned char stopsw = 0;        // 0 : NORMAL   / 1 : STOP
static unsigned char stop = 0;
static unsigned char initphase = (CHATTER_DELAY*5); // 初期化中フラグ：このフラグがONの間はパルスを出力しない

#ifdef SLOWCTL
#define DEC_OFF         ((unsigned char)0)
#define DEC_NORMAL      ((unsigned char)1)
#define DEC_FORWORD     ((unsigned char)2)
#define DEC_REVERSE     ((unsigned char)3)
static unsigned char decmode = 0;       // 0 : RA MODE  / 1 : DEC MODE / 2 : DEC FORWORD / 3 : DEC REVERSE
#endif

static unsigned short tcnt = 0;         // 設定対象タイマカウンタ値
static unsigned char next_TMR1L = 0;    // 設定対象タイマカウンタ値下位8ビット
static unsigned char next_TMR1H = 0;    // 設定対象タイマカウンタ値上位8ビット
static int phase_inc = 0;               // パルス出力時の位相変化加算値 0:stop/1:forword/-1:reverse
static unsigned char custom_tcnt_set = OFF; // EEPROMへのタイマカウンタ値設定動作状態

/*--------------------------------
    OUTPUT PULSE PHASE DEFINE
  --------------------------------*/
/* 1-2相励磁用パルステーブル */
static unsigned char pulse_tbl_12[] = {
                            0b00001001,
                            0b00001000,
                            0b00001100,
                            0b00000100,
                            0b00000110,
                            0b00000010,
                            0b00000011,
                            0b00000001
                            };
/* 2-2相励磁用パルステーブル：1-2相励磁クロック基準なので8パターン定義*/
static unsigned char pulse_tbl_22[] = {
                            /*      __*/
                            /*0000ABAB*/
                            0b00001001,
                            0b00001001,
                            0b00001100,
                            0b00001100,
                            0b00000110,
                            0b00000110,
                            0b00000011,
                            0b00000011
                            };
static unsigned char * pulse_ptbl[] = { pulse_tbl_12,pulse_tbl_22 };
unsigned char * pulse_tbl;

/*--------------------------------
    TIMER CNT DEFINE
  --------------------------------*/
static const unsigned short timer_tbl0[] = {
    /*====== GOTO MARK-X STANDARD MORTOR ======*/
    RATCNT(STAR_RATE,STD_PULSE),
    RATCNT(KINGS_RATE,STD_PULSE),
    RATCNT(SUN_RATE,STD_PULSE),
    RATCNT(MOON_RATE,STD_PULSE),
    RATCNT(STAR_HALF_RATE,STD_PULSE),
    RATCNT(STAR_RATE,STD_PULSE)/HSPEED, // DEC MODE
    0,                                  // CUSTOM MODE
    0                                   // REMOTE MODE
};
static const unsigned short timer_tbl1[] = {
    /*====== GOTO MARK-X DIRECT MORTOR ======*/
    RATCNT(STAR_RATE,DIRECT_PULSE),
    RATCNT(KINGS_RATE,DIRECT_PULSE),
    RATCNT(SUN_RATE,DIRECT_PULSE),
    RATCNT(MOON_RATE,DIRECT_PULSE),
    RATCNT(STAR_HALF_RATE,DIRECT_PULSE),
    RATCNT(STAR_RATE,DIRECT_PULSE)/HSPEED,  // DEC_MODE
    0,                                  // CUSTOM_MODE
    0                                   // REMOTE_MODE
};
static const unsigned short * timer_ptbl[] = { timer_tbl0,timer_tbl1 };
unsigned short const * timer_tbl;

```

```
/*----------------------------------
    get timer counter
  ----------------------------------*/
unsigned short timer_counter(unsigned char mode)
{
    return timer_tbl[mode];
}

#if 1
/* マクロ直接の方がコードサイズが小さい */
/*====================================
    EEPROMデータ読み込み
  ====================================*/
unsigned char eepread(unsigned char add)
{
    if( sizeof(eepdata)>add ) {
        return eepdata[add];
    }
    else {
        return EEPROM_READ(add);
    }
}
#endif

/*====================================
    データ書き込み確認、リトライ処理付
    EEPROMデータ書き込み処理
  ====================================*/
void eepwrite(unsigned char add,unsigned char data)
{
    unsigned char dt = eepread(add);
    unsigned char retry = 5;
    while( (dt != data) && retry ) {
        EEPROM_WRITE(add,data);
        dt = EEPROM_READ(add);
        retry--;
    }
    if( sizeof(eepdata)>add ) {
        eepdata[add] = dt;
    }
}

/*========================================
        sw read
  ========================================*/
#define SWDELAY         10
static unsigned char swport = 0;
unsigned char read_sw(int type)
{
    static unsigned char same = CHATTER_DELAY;
    static unsigned char swport2 = 0;
    static unsigned char scanon = 0;    //チャタリング除去スキャン中フラグ
    unsigned char swport1;

    /*================================
        DIRECT PORT CONTROL
      ================================*/
    __delay_ms(SWDELAY);    /* wait for loop */

    //ポート全ビットを負論理入力
    swport1 = ~PORTB;           // 76 543  210
                                // xx MODE xxSTOP
                                // MODE = ROTARY SW
    swport1 >>= 3;

#ifdef ROTARY_COMP          // コンプリメンタリタイプスイッチ
    swport1 = ~swport1;
#endif

    swport1 &= 0x07;

    //負論理ポート入力
    if( 0==IN_DIR )     swport1 |= DIR_MASK;
    if( 0==IN_STOP )    swport1 |= STOP_MASK;
    if( 0==IN_SPEED )   swport1 |= SPEED_MASK;

    if( -1 == type ) {
        /* チャタリングを気にせずに即座に値を返す */
        swport = swport1;
    }
    else if( scanon ) {
        if( swport1==swport2 ) {
            // 同じ値がn回続いたらチャタリング影響なしとみなして値を確定
            if( same ) same--;
            if( !same ) {
                swport = swport1;
                scanon = 0;
            }
        }
        else {
            // not same
            same = CHATTER_DELAY;
        }
    }
    else if( swport != swport1 ) {
        scanon = 1;
        same = CHATTER_DELAY;
    }
    swport2 = swport1;      /* 最新のポートデータを保存 */

    return swport;
}

/*====================================
    REMOTE MODE TIMER COUNTER GET
  ====================================*/
unsigned short remote_timer_counter(unsigned char * reverse,unsigned char * stop,unsigned char *highspeed)
{
    *reverse = OFF;
    *stop = OFF;
    *highspeed = OFF;
    return 100000;
}

/*====================================
    TIMER1 INTERRUPT ROUTINE
  ====================================*/
#ifdef SLOWCTL
#define DEC_DELAY_CNT   10
#define RA_DELAY_CNT    10
#define SLOW_STEP       3
#define SLOW_DELAY_CNT  20
#endif
void timer1_isr(void)
{
```

```
376:     static unsigned char phase = 0;
377:     static unsigned char outpulse = 0;
378: 
379:     /* OUTPUT PULSE(なるべく単純な処理で即座にパルスを出力する) */
380: #if defined(MODE_INPUT_TEST)
381:     PORTA = swport;
382: #elif defined(INPUT_TEST)
383:     PORTA = swport>>3;
384: #else
385:     PORTA = outpulse;
386: #endif
387: 
388:     /* TMR1 REGISTER SETTING */
389:     TMR1L = next_TMR1L;
390:     TMR1H = next_TMR1H;
391:     TMR1IF = 0;     /* clear interrupt flag */
392: 
393:     /* モーター出力位相更新 */
394:     phase += phase_inc;
395:     phase %= sizeof(pulse_tbl_12);
396:     outpulse = pulse_tbl[phase];
397: }
398: 
399: /*============================
400:     common interrupt routine
401:   ============================*/
402: void interrupt isr(void)
403: {
404:     /* TIMER 1 Interrupt ? */
405:     if( TMR1IF ) {
406:         timer1_isr();
407:         //TMR1IE = 1;       /* TIMER1 counter enable */
408:     }
409: }
410: 
411: /*============================
412:     setup timer counter
413:   ============================*/
414: void setup_tcnt(unsigned short tcntset)
415: {
416:     next_TMR1L = LDATA(-tcntset);
417:     next_TMR1H = HDATA(-tcntset);
418: }
419: 
420: /*============================
421:     initialize
422:   ============================*/
423: unsigned char initialize(void)
424: {
425:     unsigned char readsw;
426:     unsigned char ii;
427:     unsigned char modesw;       // 0:STAR/1:KINGS/2:SUN/3:MOON/4:STAR_HALF/5:DEC/6:CUSTOM/7:REMOTE
428: 
429:     /* initialize */
430:     di();
431:     PORTA = INIT_PORTA;
432:     PORTB = INIT_PORTB;
433:     CMCON = SET_CMCON;
434:     TRISA = SET_TRISA;
435:     RBPU = 0;               // PORTB internal pull-up enable
436:     TRISB = SET_TRISB;
437:     T1CON = SET_T1CON;
438: 
439:     __delay_ms(5);
440:     readsw = read_sw(-1);
441:     modesw = IN_MODE_SW(readsw);
442:     tcnt = timer_counter(modesw);
443:     setup_tcnt(tcnt);
444: 
445:     PEIE = 1;               /* peripheral interrupt enable */
446:     ei();
447: 
448:     eeplock = EEPROM_READ(EEP_LOCK_IDX);
449:     motor_type = EEPROM_READ(EEP_MOTORTYPE_IDX);
450:     reiji_mode = EEPROM_READ(EEP_REIJI_IDX);
451: 
452:     if(eeplock) {
453:         /* EEPROMとRAMデータに違いがあればRAMを書き換える */
454:         for(ii=0;ii<sizeof(eepdata);ii++)
455:         {
456:             unsigned char dt = EEPROM_READ(ii);
457:             eepdata[ii] = dt;
458:         }
459:     }
460:     else {
461:         /* EEPROMとRAMデータに違いがあればEEPROMを書き換える */
462:         for(ii=0;ii<sizeof(eepdata);ii++)
463:         {
464:             unsigned char dt = EEPROM_READ(ii);
465:             if( dt!=eepdata[ii] ) {
466:                 eepwrite(ii,eepdata[ii]);
467:             }
468:         }
469:     }
470: 
471:     di();
472: 
473:     /* setup timer counter table */
474:     timer_tbl = timer_ptbl[motor_type];
475: 
476:     /* setup pulse phase table */
477:     pulse_tbl = pulse_ptbl[reiji_mode];
478: 
479:     TMR1L = next_TMR1L;
480:     TMR1H = next_TMR1L;
481:     TMR1IE = 1;             /* TIMER 1 interrupt enable*/
482:     TMR1IF = 0;             /* TIMER1 interrupt flag clear */
483:     TMR1ON = 1;             /* TIMER1 ON */
484: 
485:     ei();
486: 
487:     return readsw;
488: }
489: 
490: /*======================
491:     setup mode
492:   ======================*/
493: void modeset(unsigned char readsw,unsigned char mode)
494: {
495:     if( MODE_REMOTE==mode ) {
496:         /* REMOTE MODE */
497:         tcnt = remote_timer_counter(&reverse,&stop,&highspd);
498:         setup_tcnt(tcnt);
499:     }
500:     else {
```

```
        /* NORMAL RA MODE or DEC MODE*/

        /* PUSH SW OPERATION */
        reversesw = IN_DIR_SW(readsw);
        stopsw = IN_STOP_SW(readsw);
        highspdsw = IN_SPEED_SW(readsw);

        reverse = reversesw;
        if( MODE_DEC==mode ) {
            /* DEC MODE */
            /* TIMER COUNT GET */
            tcnt = timer_counter(mode);
            setup_tcnt(tcnt);

            highspd = OFF;
            stop = ON;
            if( stopsw ) {
                if( motor_type ) {
                    reverse = !reversesw;
                }
                stop = OFF;
            }
            else if( highspdsw ) {
                if( !motor_type ) {
                    reverse = !reversesw;
                }
                stop = OFF;
            }
        }
        else {
            /* RA MODE */
            if( !motor_type ) {
                /* GOTO STANDARD */
                /* 平ギアで逆転しているので正転の時が逆転 */
                reverse = !reversesw;
            }
            stop = stopsw;
            highspd = highspdsw;

            /* TIMER COUNT GET */
            if( MODE_CUSTOM==mode ) {
                // CUSTOM MODEの時はEEPROMにカウンタ値が設定してあるものとする
                tcnt = eepread(EEP_CUSTOM_TIMER_H);
                tcnt <<= 8;
                tcnt |= eepread(EEP_CUSTOM_TIMER_L);
            }
            else {
                tcnt = timer_counter(mode);
            }
            if(highspd) {
                tcnt /= HSPEED;
            }
            setup_tcnt(tcnt);
        }
    }
}

/*-----------------------------------
        option change
  -----------------------------------*/
void option_change(unsigned char readsw,unsigned char modesw)
{
#ifdef OPTION_CHANGE
    static unsigned char last_modesw = MODE_STAR_RATE;
    static unsigned char setcnt = 0;
    static unsigned char latch = 0;
    static unsigned short custom_tcnt;
    unsigned char stopsw = IN_STOP_SW(readsw);
    unsigned char speedsw = IN_SPEED_SW(readsw);

    if( custom_tcnt_set ) {
        /* カスタムモード、タイマカウンタ設定中 */
        if( OFF==latch && ON==stopsw ) {
            // 入力ラッチ
            // スピードスイッチの状態を1ビットずつ取り込む
            custom_tcnt <<= 1;
            custom_tcnt |= speedsw;
            setcnt++;
            if( setcnt>=16 ) {
                //カウンタ取り込み完了
                eepwrite(EEP_CUSTOM_TIMER_H,HDATA(custom_tcnt));
                eepwrite(EEP_CUSTOM_TIMER_L,LDATA(custom_tcnt));
                custom_tcnt_set = OFF;
            }
        }
        latch = stopsw;
    }
    else if( modesw==last_modesw ) {
        /* モードの変化がなければ何もせずに抜ける */
        return;
    }
    else if( MODE_DEC==modesw && stopsw && speedsw ) {
        /* 励磁モード切替 */
        reiji_mode++;
        reiji_mode %= 2;
        eepwrite(EEP_REIJI_IDX,reiji_mode);

        /* setup pulse phase table */
        di();
        pulse_tbl = pulse_ptbl[reiji_mode];
        ei();
    }
    else if( MODE_CUSTOM==modesw && stopsw && speedsw ) {
        /* モータータイプ切替 */
        motor_type++;
        motor_type %= 2;
        eepwrite(EEP_MOTORTYPE_IDX,motor_type);

        /* setup timer counter table */
        di();
        timer_tbl = timer_ptbl[motor_type];
        ei();
    }
    else if( MODE_REMOTE==modesw && stopsw && speedsw ) {
        /* カスタムモード時タイマカウンタ設定操作モード遷移 */
        custom_tcnt_set = ON;
        custom_tcnt = 0;
        setcnt = 0;
        latch = stopsw;
    }
    last_modesw = modesw;
#endif
}

/********************
```

```
626:     MAIN
627:  ********************/
628: void main(void)
629: {
630:     unsigned char last_sw = initialize();
631:     modeset(last_sw,IN_MODE_SW(last_sw));
632:     while(1)
633:     {
634:         /* sw data read */
635:         unsigned char readsw = read_sw(0);
636:         unsigned char modesw = IN_MODE_SW(readsw);
637:
638: #ifdef EEPTEST
639:         eepwrite(EEP_SW_IDX,readsw);
640:         eepwrite(EEP_MODESW_IDX,modesw);
641: #endif
642:
643:         if( readsw!=last_sw ) {
644:             option_change(readsw,modesw);
645:             modeset(readsw,modesw);
646:             last_sw = readsw;
647:         }
648:         // 初期化ディレイタイマカウント
649:         if( initphase ) {
650:             initphase--;
651:         }
652:
653:         if ( custom_tcnt_set ) {
654:             /* カスタムモードタイマカウンタ設定操作 */
655:             // 断続的にパルス出力を行う
656:             static unsigned char blink = 0;
657:             blink++;
658:             if( 0==blink ) {
659:                 phase_inc++;
660:                 phase_inc &= 1;
661:             }
662:         }
663:         else if( initphase || stop ) {
664:             /* 初期起動してしばらくは設定値読み取りが安定していないのでパルスを更新出力しない */
665:             /* stop状態の時もパルス更新出力しない */
666:             phase_inc = 0;
667:         }
668:         else {
669:             /* パルスモータ位相更新設定 */
670:             if( !reverse ) {
671:                 /* forword */
672:                 phase_inc = 1;
673:             }
674:             else {
675:                 /* reverse */
676:                 phase_inc = -1;
677:             }
678:         }
679:     }
680: }
681:
682: #if 0
683:     /* RA && STOP時とDECモードノーマル状態またはdec_delayが0でない間停止 */
684:     if( (MODE_DEC!=modesw) && (stop || ra_delay) ) {
685:         return;
686:     }
687:     else if((MODE_DEC==modesw) && (dec_delay || (DEC_NORMAL==decmode))) {
688:         return;
689:     }
690:
691: #endif
692: #ifdef SLOWCTL
693:     /* DEC MODE時は急激な反転を避けるために必ず停止を一定時間挟む */
694:     static unsigned char dec_delay = 0;
695:     static unsigned char ra_delay = 0;
696:     static unsigned char decmode_last = 0xff;
697:     static unsigned char direction_last = 0xff;
698:     static unsigned char slowstep = 0;
699:     static unsigned char slow_delay = 0;
700:
701:     if(DEC_NORMAL<=decmode) {
702:         if( decmode==decmode_last ) {
703:             /* dec_modeに変化がなければdelayタイマカウントする */
704:             if( dec_delay ) dec_delay--;
705:             if( !dec_delay && slow_delay ) {
706:                 tcntset *= SLOW_STEP;
707:                 slow_delay--;
708:             }
709:         }
710:         else if(dec_delay) {
711:             /* dec_modeに変化があってもdelayタイマカウントが残っていればそのまま計時 */
712:             dec_delay--;
713:         }
714:         else {
715:             /* dec_modeが変ったので停止を一定区間挟んで切り替える */
716:             if( (DEC_NORMAL!=decmode_last) && (DEC_NORMAL!=decmode)) {
717:                 /* FORWORDとREVERSE切り替えなのでSTOPを挟むためにディレイカウントする */
718:                 dec_delay = DEC_DELAY_CNT;
719:                 slow_delay = SLOW_DELAY_CNT;
720:             }
721:             else {
722:                 /* DEC_NORMAL時はもともと停止なのでディレイカウントしない */
723:                 dec_delay = 0;
724:             }
725:             decmode_last = decmode;
726:         }
727:     }
728:     else {
729:         /* RAモード時でもFORWORDとREVERSEの切り替え時には停止を挟むようにする */
730:         if( direction_last != reverse ) {
731:             ra_delay = RA_DELAY_CNT;
732:             slow_delay = SLOW_DELAY_CNT;
733:             direction_last = reverse;
734:         }
735:         else {
736:             if( ra_delay ) ra_delay--;
737:             if( !ra_delay && slow_delay ) {
738:                 tcntset *= SLOW_STEP;
739:                 slow_delay--;
740:             }
741:         }
742:     }
743: #endif
```