ภาคผนวก ก

เปรียบเทียบการวัดความเร็ว

ตาราง ก-1 ได้แสดงการเปรียบเทียบผลการวัดความเร็ว ที่ได้จากวิธีการวัดที่ใช้ในการทดลองครั้งที่ 1 และครั้งที่ 2 กับวิธีการวัดความเร็วในการทดลองที่ 3 เมื่อทำการวัดความที่ที่ได้จากเครื่องกำเนิดสัญญาณความถี่ 1 KHz ซึ่งแสดงให้เห็นว่าที่ความถี่หรือความเร็วเดียวกันวิธีที่ได้ทำการปรับปรุงความเร็วในการวัดนั้นจะวัดได้ค่าที่น้อยกว่า ดังนั้นในการควบคุมความเร็วของระบบจะเสมือนว่าต้องควบคุมที่ความเร็วสูงกว่าระบบควบคุมแบบเก่าจึงทำให้ต้องมีการลดค่าความเร็วอ้างอิงลงเพื่อให้มอเตอร์ตัวหลักสามารถทำงานได้ได้โดยไม่มีการเสริมกำลัง

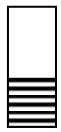
ตาราง ก-1 เปรียบเทียบผลการวัดความเร็วระหว่างการวัดที่ใช้ในการทดลองครั้งที่ 1 และครั้งที่ 2 กับวิธีที่ใช้ในการทดลองครั้งที่ 3

ความถี่ที่วัดได้ก่อนปรับปรุง(Hz)	ความถี่ที่วัดได้เมื่อทำการปรับปรุง(Hz)
1000	990
1000	990
1000	996
1000	990
995	990
995	990
995	990
995	996
1000	990
1000	996
1000	990
1000	990
995	990

ความถี่ที่วัดได้ก่อนปรับปรุง(Hz) (ต่อ)	ความถี่ที่วัดได้เมื่อทำการปรับปรุง(Hz)(ต่อ)
995	990
995	990
995	990
1000	990
1000	990
1000	990
1000	990
995	990
995	990
995	996
995	990
1000	990
1000	990
1000	990
995	996
995	990
เฉลี่ย 998±2	เฉลี่ย 991± 2

ภาคผนวก ข
การทำงานของเครื่องมือ

การทำงานของระบบการเสริมกำลังในงานวิจัยนี้ สามารถทำงานได้สองโหมดการทำงาน คือแบบเชื่อมต่อกับคอมพิวเตอร์ และแบบทำงานโดยลำพัง (Stand Alone) ไม่เชื่อมต่อกับคอมพิวเตอร์ซึ่งในกาเลือกโหมดการทำงานจะต้องทำการเลือกจากสวิทซ์ ดังภาพประกอบ ข - 1



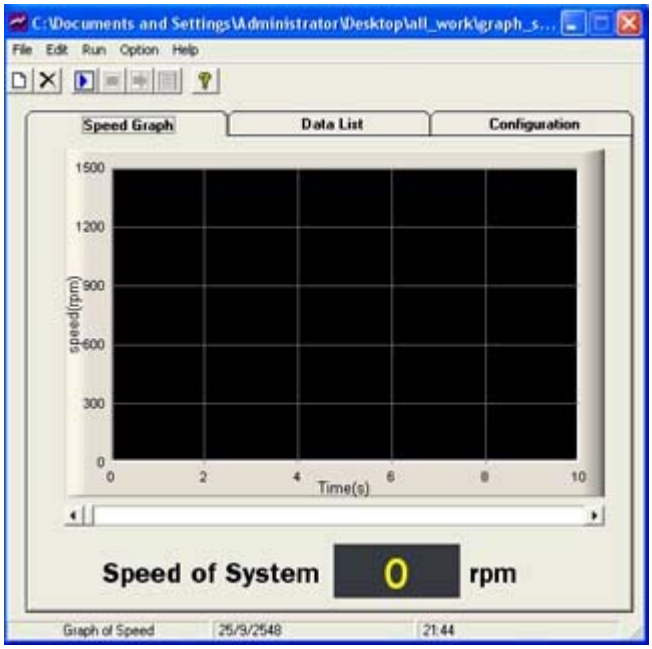
ภาพประกอบ ข-1 แสดงสวิทซ์เลือกโหมดในการทำงานของส่วนควบคุม

โหมดเชื่อมต่อคอมพิวเตอร์

ในการใช้งานโหมดนี้ผู้ใช้จะต้องทำการปรับสวิทซ์เพื่อทำการเลือกโหมด **Computer** ดังภาพประกอบ ข-2

ภาพประกอบ ข-2 แสดงการเลือกโหมด เชื่อมต่อคอมพิวเตอร์

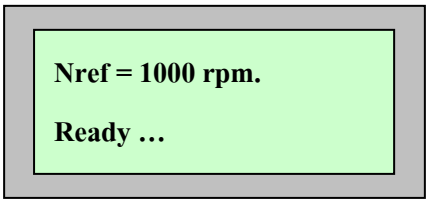
ในโหมดนี้จะเป็นโหมดที่ใช้ในการทดลองเพื่อทำการทดสอบโปรแกรมและเก็บข้อมูลความเร็วของระบบ ซึ่งจะใช้คอมพิวเตอร์เชื่อมต่อกับเครื่องมือผ่านพอร์ตอนุกรม ดังนั้นในโหมดนี้จำเป็นจะต้องมีโปรแกรมที่ทำงานบนคอมพิวเตอร์เพื่อทำการเก็บข้อมูลดังกล่าว โปรแกรมที่ใช้ผู้ทำการวิจัยได้ทำการพัฒนาขึ้นเพื่อใช้ในงานวิจัยนี้ดังภาพประกอบ ข-3



ภาพประกอบ ข-3 แสดงโปรแกรมที่ใช้ในการรับค่าความเร็วของระบบ

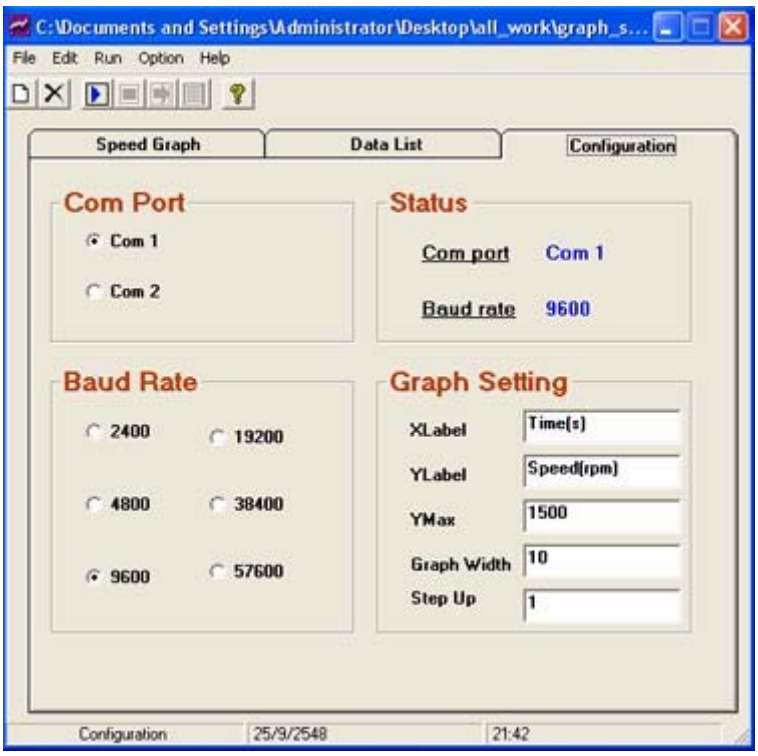
การใช้งานโปรแกรมมีขั้นตอนดังนี้

1. ทำการเชื่อมต่อเครื่องมือกับคอมพิวเตอร์ผ่านพอร์ตอนุกรมและทำการเปิดเครื่อง จะมีข้อความ แสดงความเร็วอ้างอิง และสถานะว่าพร้อมหรือไม่พร้อมที่จอแสดงผลแบบ LCD ค้างอยู่เพื่อรอสัญญาณเริ่มต้นจากคอมพิวเตอร์ดังภาพประกอบ ข-4



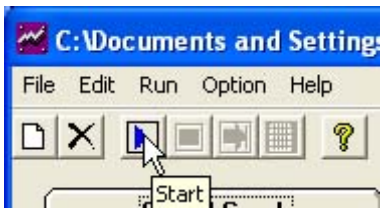
ภาพประกอบ ข-4 แสดงข้อความแสดงความพร้อมเมื่อทำงานในโหมดเชื่อมต่อกับคอมพิวเตอร์

2. ทำการกำหนดค่าต่างๆ เพื่อให้โปรแกรมบนคอมพิวเตอร์สอดคล้องกับการทำงานของฮาร์ดแวร์ เช่นกำหนดพอร์ต และความเร็วในการรับส่งข้อมูลและกำหนดค่าต่างๆ ของการแสดงกราฟ ดังภาพประกอบ ข-5



ภาพประกอบ ข-5 แสดงการกำหนดค่าต่างๆ ของโปรแกรมเพื่อให้สอดคล้องกับฮาร์ดแวร์

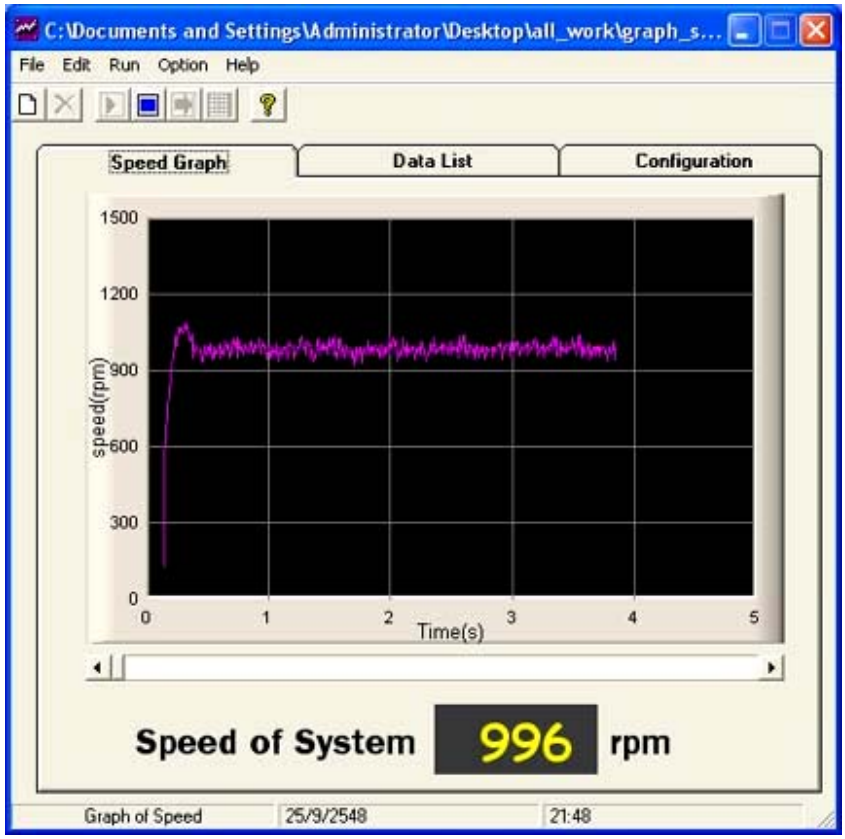
3. ทำการกดปุ่ม *Start* เพื่อทำการส่งสัญญาณเริ่มต้นการทำงานไปยังเครื่องมือ



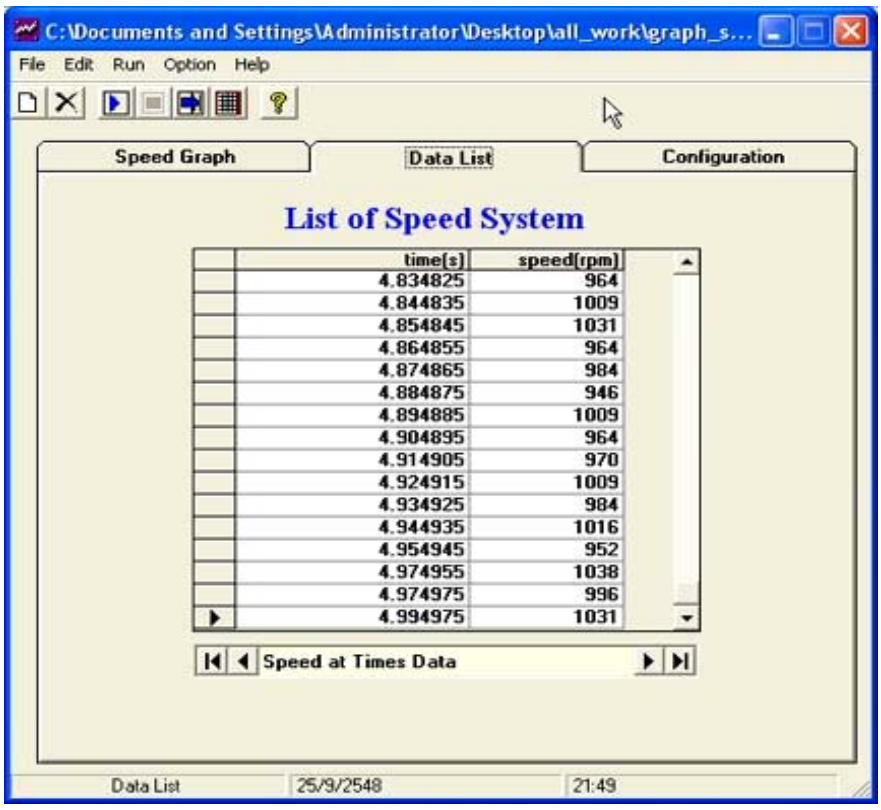
ภาพประกอบ ข-6 แสดงการสั่งให้ระบบเริ่มทำงาน

4. เมื่อทำการส่งสัญญาณเริ่มต้นเครื่องมือจะทำการส่งข้อมูลมายังโปแกรม และทำการพล็อตกราฟแบบเวลาจริง โดยข้อมูลที่ได้จะทำการจัดเก็บไว้ในฐานข้อมูลชั่วคราวที่โปรแกรมทำการสร้างขึ้น โดยผู้ใช้สามารถบันทึก และวิเคราะห์ข้อมูลดงกล่าวโดยการส่งขอมูลดังกล่าว

ไปยังโปรแกรมประเภท ตารางคำนวณ

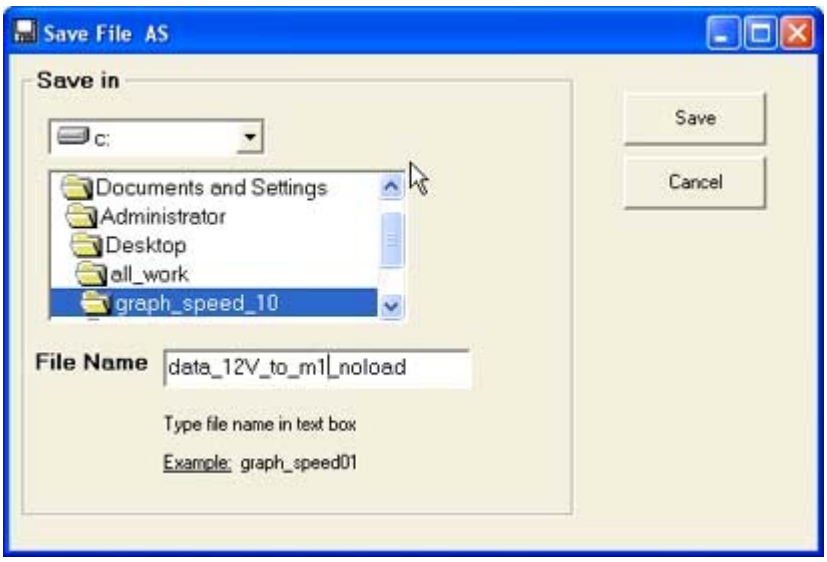


ภาพประกอบ ข-7 แสดงกราฟขณะทำการรันเครื่องมือ



ภาพประกอบ ข-8 แสดงข้อมูลที่ทำการรับมาเก็บไว้ในฐานข้อมูลชั่วคราว

ภาพประกอบ ข-9 แสดงวิธีการบันทึกกราฟเป็นรูปภาพ



ภาพประกอบ ข-10 บันทึกภาพที่ได้จากการพล็อตกราฟ

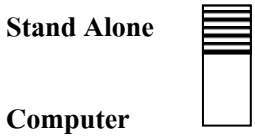
ภาพประกอบ ข-11 วิธีการส่งข้อมูลความเร็วและเวลาที่ได้ไปยังโปรแกรมประเภท ตารางคำนวณ

	A	B	C
1	**time(s)**	**speed(rpm)**	
2	0.15	130	
3	0.15015	561	
4	0.16015	647	
5	0.17016	753	
6	0.18017	814	
7	0.20018	880	
8	0.2102	923	
9	0.22021	964	
10	0.23022	1009	
11	0.24023	1038	
12	0.25024	1003	
13	0.26025	1073	
14	0.27026	1066	
15	0.28027	1052	
16	0.29028	1066	

ภาพประกอบ ข-12 แสดงข้อมูลในโปรแกรมตารางคำนวณ ที่ส่งมาจากโปรแกรมรับค่าความเร็วของระบบ

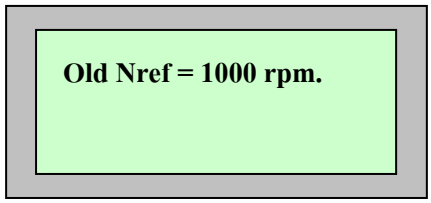
โหมดทำงานแบบลำพัง

โหมดนี้จะเป็นโหมดที่จะนำไปใช้งานจริงๆ ซึ่งอาจไม่ต้องมีการเชื่อมต่อกับคอมพิวเตอร์ ในการเลือกใช้โหมดนี้นั้นผู้ใช้งานจะต้องทำการเลือกสวิทซ์ดังภาพประกอบ ข-13



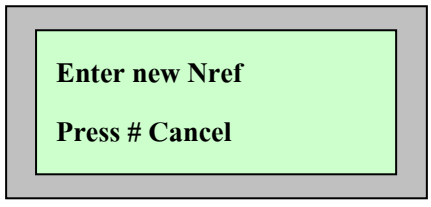
ภาพประกอบ ข-13 การเลือกโหมดทำงานโดยไม่เชื่อมต่อกับคอมพิวเตอร์

เมื่อทำการเลือกโหมดนี้โปรแกรมจะแจ้งให้ผู้ใช้งานสามารถเปลี่ยนแปลงความเร็วอ้างอิงได้ โดยจะมีการรายงานความเร็วอ้างอิงปัจจุบันที่ทำการเก็บไว้ในหน่วยความจำแบบ อีอีพรอมดังภาพประกอบ ข-14



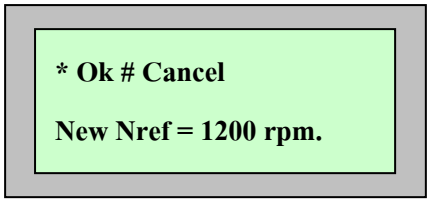
ภาพประกอบ ข-14 แสดงข้อความแจ้งความเร็วอ้างอิงปัจจุบัน

หลังจากนั้น 6 วินาที ระบบจะแสดงข้อความดังภาพประกอบ ข-15 โดยสามารถกด # คีย์สวิทซ์ของกล่องควบคุมเพื่อยกเลิกการตั้งค่าความเร็วอ้างอิง ระบบจะใช้ความเร็วอ้างอิงค่าเดิมในการควบคุม



ภาพประกอบ ข-15 แสดงข้อความแจ้งความให้ป้อนค่าความเร็วอ้างอิง

ถ้าผู้ใช้ทำการป้อนค่าความเร็วอ้างอิงใหม่ระบบจะแสดงตัวเลขที่ผู้ใช้ป้อนโดยโปรแกรมจะทำการถามเพื่อยืนยันค่าที่ทำการป้อนตัวอย่าง เช่นต้องการป้อนค่าใหม่ 1200 รอบต่อนาทีระบบจะแสดงข้อความดังภาพประกอบ ข-16

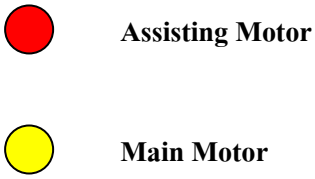


ภาพประกอบ ข-16 แสดงข้อความเพื่อยืนยันการเปลี่ยนความเร็วอ้างอิง

โดยถ้าต้องการยืนยันการเปลี่ยนแปลงให้ทำการกด * บนคีย์สวิทซ์ของกล่องควบคุม แต่ถ้าต้องการยกเลิกค่าที่ทำการป้อนผู้ใช้สามารถกด # บนคีย์สวิทซ์ของกล่องควบคุมเพื่อทำการยกเลิกขณะทำการรันโดยใช้โหมดนี้ระบบจะทำการแสดงความเร็วของระบบบนจอแสดงผลแบบแอลซีดีตลอดเวลา

สถานะทำงานของมอเตอร์แต่ละตัว

ระบบควบคุมจะแสดงสถานะทำงานของมอเตอร์ทั้งสองตัวเพื่อให้เราทราบว่าว่าตอนนี้ระบบใช้แหล่งพลังงานหรือกำลังใช้มอเตอร์ตัวไหนการขับเคลื่อนภาระ โดยจะใช้ ไดโอดเปล่งแสงแสดงสถานะทำงานของมอเตอร์แต่ละตัว



ภาพประกอบ ข-17 แสดงสถานะทำงานของมอเตอร์แต่ละตัว

ภาคผนวก ค

โปรแกรมฝั่งไมโครคอนโทรลเลอร์

```
#include<18f458.h>
#use delay(clock=40000000)
#fuses H4,NOPROTECT,NOWDT,NOBROWNOUT,NOPUT,NOLVP
#use rs232(baud=9600,xmit=pin_c6,rcv=pin_c7)
//=====================================================================
#define MOTOR_1 PIN_C1
#define MOTOR_2 PIN_C5
#define MINRPM_1 0.1   // percen of main motor rpm less than nref
long const INIT_DUTY1=1000;
long const INIT_DUTY2 = 900;  // if nref = 1200 INIT_DUTY2 = 1000
//=====================================================================
long set_nref(void);
void init_motor(void);
long gets_speed(void);
void set_speed_m1(char s);
void set_speed_m2(char s);
signed long determine_err(long nref);
signed int fuzzy_1(signed long err,signed long errpre );
//======================== function for check volt ============================
float read_volt_1();
float read_volt_2();
int1 get_status(float volt);
void check_two_power();
int1 check_power_m1();
int1 check_power_m2();
```

```
//setup_adc_ports(A_ANALOG);
//setup_adc(ADC_CLOCK_INTERNAL);
//==================== config and define for lcd ======================
// As defined in the following structure the pin connection is as follows:
//////////////////////////////////////////////////////////////////////////
// microcotroller Connect to LCD
//    E0    --------->    enable
//    E1    --------->    rw
//    E2    --------->    rs

//    D0    --------->    D4
//    D1    --------->    D5
//    D2    --------->    D6
//    D3    --------->    D7
//////////////////////////////////////////////////////////////////////////
#define lcd_type 2          // 0=5x7, 1=5x10, 2=2 lines
#define lcd_line_two 0x40    // LCD RAM address for the second line

BYTE const LCD_INIT_STRING[4] = {0x20 | (lcd_type << 2), 0xc, 1, 6};
                    // These bytes need to be sent to the LCD
                    // to start it up.
                    // for 2 lines lcd: {0x0a,0xc,1,6}
                    // for 5x10 lcd:   {0x06,0xc,1,6}
                    // for 5x7 lcd:      {0x20,0xc,1,6}
                    // The following are used for setting
                    // the I/O port direction register.

//******************** lcd command port  ************************************
struct lcd_command{
   char enable:1;
```

```
  char read_write:1;      //rw
  char data_command:1;    //rs
  };struct lcd_command lcd_command_port;
#locate lcd_command_port=0xf84         // use port E for controll bit
//-------------------------------------------------
struct lcd_data{
  char data:4;
  };
struct lcd_data lcd_data_port;
#locate lcd_data_port=0xf83            // use port D for data bit

//****************  "COMMAND / DATA definition ***************************
#define COMMAND0 0
#define DATA1 1

//************  "READ / WRITE definition  ************************************
#define WRITE0 0   //write data to lcd
#define READ1 1    //read  data from lcd

//************  "E_UP / E_DOWN definition  ***********************************
#define E_DOWN 0
#define E_UP 1
//************************   clear LCD  ***************************************
#define clear_lcd   lcd_send_byte(0x1,COMMAND0);//clear lcd display

//*****************************************************************************
char TRISD =0;  //d4 - d7 output
//*****************************************************************************
////////////////////////////////////////////////////////////////////////////////
//==========================function for lcd ================================
```

```
////////////////////////////////////////////////////////////////////////////
void set_tris_lcd_read(void);
void set_tris_lcd_write(void);
char lcd_read_byte(void);
void lcd_write_nibble(char temp_wr);
void lcd_send_byte(char data , char data_or_command);
void lcd_gotoxy( char location, char line);
void lcd_init(void);
void lcd_putc( char c);
char lcd_getc( char location, char line);
//***************************************************************************
//====================== define for keypad ============================
//***************************************************************************
#byte PORTB = 0xF81
int1 keypressed;
char code[12]={0xe5,0x73,0x75
              0x76,0xb3,0xb5
              0xb6,0xd3,0xd5
              0xd6,0xe3,0xe6};
long data[4];
long dat;
//***************************************************************************
//==================== function of keypad and EEPROM ==================
//***************************************************************************
void WRITE_LONG_EEPROM(long int n, long data);
long READ_LONG_EEPROM(long int n);
void beep(void);
void move_digi(void);
void clear_data(void);
char get_kbd(void);
```

```
//
//*****************************************************************************
//============================= main function ================================
//*****************************************************************************
//
int1 Not_start=1;
#int_rda
void rs232_isr()
{
  if(Not_start)
  {
    Not_start = 0;
    getch();
  }
  else
  {
    Not_start=1;
    getch();
    reset_cpu();
  }

}
void main()
{
//==================  declaration variable ================================

  long nref;
  signed long err,errch,errpre;
  signed int ch_duty;
  long duty1,duty2;
```

```
   int temp;
   signed int temp1;
   int1 m1_stoped =0;
   int1 m2_stoped =0;
   int1 stoped=0;       //flag for chek stop assisted
   int1 first_one = 1;
   int1 first_two = 1;
   int1 main_power_lose = 0;
   int count;
   int1 assist_command =0;
   long n_count;
   long count_loop;
   int1 ready=0;

//*****************************************************************************
//*****************************************************************************
   enable_interrupts(GLOBAL);
   enable_interrupts(INT_RDA);
   //setup_adc_ports(A_ANALOG);
   //setup_adc(ADC_CLOCK_INTERNAL);
   lcd_init();
   init_motor();
   delay_ms(500);
   duty1 = INIT_DUTY1;
   duty2 = INIT_DUTY2;
   if(!input(PIN_C4))   // stand Alone
   {
      nref = set_nref();   //input speed reference from keypad
   }
   else           // Connect to Computer
```

```
 {
   nref = READ_LONG_EEPROM(0);
   printf(lcd_putc,"Nref=%lu rpm",nref);
   delay_ms(500);
   printf(lcd_putc,"\nready....");
 }
//nref = set_nref();   //input speed reference from keypad
 temp=MINRPM_1*nref;            // for check to assisted
 temp1 = (-0.1)*(nref);        // for check overshoot
 errpre = nref;         //initial errpre = nref - 0
 Not_start =1;
 if(input(PIN_C4))
 {
   while(Not_start){}
 }
 delay_ms(200);
 check_two_power();   //check  power supply  //////////////////////////
 m1_stoped = check_power_m1();
 set_speed_m1(INIT_DUTY1);
 set_speed_m2(INIT_DUTY2);
 if(!m1_stoped)   //not stop main motor
  {
   main_power_lose = 0;
   output_high(MOTOR_1);  //enable main motor
   output_low(MOTOR_2);   //disable assist motor
  }
  else
  {
   main_power_lose = 1;
  }
```

```
//
while(true)       //non stop loop
{
  if(stoped)     //flag test for stop assisted
  {
    output_low(MOTOR_2);        //stop assisted
    duty2 = INIT_DUTY2;
    set_speed_m2(INIT_DUTY2);   //reset duty2 to initial value
    stoped = 0;
  }
  if(count_loop<1000)
  {
    ready =0;
    count_loop=count_loop+1;
  }
  else
  {
    ready =1;
    printf(lcd_putc,"\f ready.... \n");
  }

  m1_stoped = check_power_m1();
  if(!m1_stoped)      //not stop main motor
  {
    main_power_lose = 0;
    if(first_one)
    {
      err = determine_err(nref);
      delay_ms(20);
      first_one = 0;
```

```
   //printf("err=%ld\n\r",err);
}

if(err!=0)
{

      ch_duty = fuzzy_1(err,errpre );
}
else
{
    //ch_duty = 0;
}
//printf("ch_duty = %d\n\r",ch_duty);
duty1 = duty1+ch_duty;
if (duty1 > 1023)
{
  duty1 = 1023;
}
//printf("duty1=%ld\n\r",duty1);
set_speed_m1(duty1);
//delay_ms(20);
errpre = err;        // importance
err = determine_err(nref);
delay_ms(20);
if((err>15)&&(err<temp)&&(ready))
  {
     n_count = n_count+1;
  }
  else
  {
```

```
            n_count =0;
          }
          if(n_count==15)
          {
            assist_command =1;
          }
          else
          {
            assist_command =0;
          }
        //printf("err=%ld\n\r",err);
      }
      else
      {
        main_power_lose = 1;
      }
      if(((err>=temp)&&(duty1==1023))||(m1_stoped)||(assist_command))//low power to support
main motor
      {
        n_count =0;
        m2_stoped = check_power_m2();
        first_two = 1;
        first_one =1;
        stoped = 1;
        while((!(err<temp1))&&(!m2_stoped)||((!m2_stoped)&&(m1_stoped)))
        {
          m2_stoped = check_power_m2();
          m1_stoped = check_power_m1();
          if((!m1_stoped)&&(main_power_lose))
          {
```

```
  main_power_lose = 0;
  break;
}
if(first_two)
{
   err = determine_err(nref);
   delay_ms(20);
   first_two = 0;
}
if(err!=0)
{

     ch_duty = fuzzy_1(err,errpre );

}
else
{
  ch_duty = 0;
}
 //printf("ch_duty = %d\n\r",ch_duty);
 duty2 = duty2+ch_duty;
 if(duty2>1023)
 {
  duty2=1023;
 }
 //printf("duty2 = %ld\n\r",duty2);
 set_speed_m2(duty2);
 errpre = err; // importance
 err = determine_err(nref);
 delay_ms(20);
```

```
            //printf("err=%ld\n\r",err);
           } //while((!(err<temp1))&&(!m2_stoped))
           //printf("stop assisted\n\r");
      } //end if
   }//end while(true)
}  // end main
////////////////////////////////////////////////////////////////////////////
//***************************************************************************
//===========function for read volt from  power suply of main motor ============
//***************************************************************************
/*float read_volt_1()
{
   int digital_data;
   float volt1;
   set_adc_channel(1);
   delay_us(50);
   digital_data = read_adc();
   //volt1 = (4.0/51.0)*adc1;
   volt1 = (4.0*digital_data)/(51.0);
   //setup_adc( ADC_OFF );
   return volt1;
}*/
//***************************************************************************
//========= function for read volt from  power suply of assist motor ===========
//***************************************************************************
/*float read_volt_2()
{
   int data2;
   float volt2;
```

```
  set_adc_channel(2);
  delay_us(50);
  data2 = read_adc();
  volt2 = (4.0/51.0)*data2;
  //setup_adc( ADC_OFF );
  return volt2;
}*/
//*****************************************************************************
//======= this function for get status power suply to each motor ===============
//*****************************************************************************
/*int1 get_status(float volt)
{
  if((volt<=7.0)||(volt>=18.2))
  {
    return 1;
  }
  else
  {
    return 0;
  }
}*/
//*****************************************************************************
//============  this function check for enable or disable all motor  ==========
//*****************************************************************************
/*void check_two_power()
{
  int1 stop_m1 = 0;
  int1 stop_m2 = 0;
  int i;
  float volt_1,volt_2;
```

```
  volt_1 = read_volt_1();
  volt_2 = read_volt_2();
 //// lcd_gotoxy(1,2);
  printf(lcd_putc,"v1=%2.1f,v2=%2.1f",volt_1,volt_2);
  stop_m1 = get_status(volt_1);
  stop_m2 = get_status(volt_2);
    //
  while((stop_m1)&&(stop_m2))
  {
    output_low(MOTOR_1);
    output_low(MOTOR_2);
    //lcd_gotoxy(1,1);
    //lcd_putc("Stop m1 & m2");
    delay_ms(1000);
    volt_1 = read_volt_1();
    volt_2 = read_volt_2();
   // lcd_gotoxy(1,2);
    //printf(lcd_putc,"v1=%2.1f,v2=%2.1f",volt_1,volt_2);
    stop_m1 = get_status(volt_1);
    stop_m2 = get_status(volt_2);
  }
}
//*****************************************************************************
//============== this function check for enable or disable main motor ==========
//*****************************************************************************
int1 check_power_m1()
{
  int1 stop_m1 = 0;
  int i;
```

```
  float volt_1,volt_2;
   volt_1 = read_volt_1();
   //volt_2 = read_volt_2();
   //lcd_gotoxy(1,2);
   //printf(lcd_putc,"v1=%2.1f,v2=%2.1f",volt_1,volt_2);
   stop_m1 = get_status(volt_1);
   //delay_ms(25);
  if(stop_m1)
  {
    output_low(MOTOR_1);
    printf(lcd_putc,"\n\rStop main motor");
    return 1;
  }
  else
  {
    printf(lcd_putc,"\f");
    output_high(MOTOR_1);
    return 0;
  }
}

//*****************************************************************************
//============  this function check for enable or disable assist motor  =======
//*****************************************************************************
int1 check_power_m2()
{
  int1 stop_m2 = 0;
  int i;
  float volt_1,volt_2;
```

```
  //volt_1 = read_volt_1();
  volt_2 = read_volt_2();
  //lcd_gotoxy(1,2);
  //printf(lcd_putc,"v1=%2.1f,v2=%2.1f",volt_1,volt_2);
  stop_m2 = get_status(volt_2);
  if(stop_m2)
  {
    output_low(MOTOR_2);
    printf(lcd_putc,"\n\rStop Assist Motor");
    return 1;
  }
  else
  {
    printf(lcd_putc,"\f");
    output_high(MOTOR_2);
    return 0;
  }
}*/
//************************************************************************
//// this function  for set speed reference via keybord and storage to EEPROM //
//************************************************************************
void check_two_power()
{
  while((input(PIN_A1))&&(input(PIN_A2)))
  {
    lcd_putc("Stop m1&m2");
    output_low(MOTOR_1);
    output_low(MOTOR_2);
    delay_ms(1000);
  }
```

```
}
int1 check_power_m1()
{
  if(input(PIN_A1))
  {
    output_low(MOTOR_1);
    return 1;
  }
  else
  {
    output_high(MOTOR_1);
    return 0;
  }
}
int1 check_power_m2()
{
  if(input(PIN_A2))
  {
    output_low(MOTOR_2);
    return 1;
  }
  else
  {
    output_high(MOTOR_2);
    return 0;
  }
}
long set_nref(void)
{
  long Nnew,Nold;
```

```
char key_in;
int1 press_keys = true;
set_tris_b(0xE0);
keypressed = 0;
Nold = READ_LONG_EEPROM(0);
printf(lcd_putc,"\fOld nref=%lu rpm\n",Nold);
delay_ms(6000);
printf(lcd_putc,"\f Enter new nref \n");
printf(lcd_putc," press # cancle ");
clear_data();
while(press_keys)
{

  key_in = get_kbd();
  if(key_in != 0xFF)
  {
    if(key_in <= 9)
    {
      move_digi();
      data[3]= key_in;
      dat = (data[0]*1000)+(data[1]*100)+(data[2]*10)+data[3];
      //clear_data();
      printf(lcd_putc,"\f* ok # cancle\n");
      printf(lcd_putc,"New nref=%lu rpm",dat);
    }
    else if(key_in == 10)
    {
      WRITE_LONG_EEPROM(0,dat);
      Nnew = READ_LONG_EEPROM(0);
      printf(lcd_putc,"\f....Comleted....\n");
```

```
            printf(lcd_putc,"nref=%lu rpm.",Nnew);
            delay_ms(3000);
            clear_data();
            dat = 0;
            press_keys = False;
          }
          else if(key_in == 11)
          {
            Nnew = READ_LONG_EEPROM(0);
            printf(lcd_putc,"\f....Cancled....\n");
            delay_ms(3000);
            printf(lcd_putc,"\fnref=%lu rpm\n",Nold);
            delay_ms(5000);
            clear_data();
            dat = 0;
            press_keys = False;
          }
       }//end if
    }//end while
    printf(lcd_putc,"\fready....");
    return Nnew;
}
////////////////////////////////////////////////////////////////////////////
// ===================== function for lcd ================================
////////////////////////////////////////////////////////////////////////////
void set_tris_lcd_read(){
    set_tris_e(0);
    set_tris_d(TRISD | 0x0f);
    }
void set_tris_lcd_write(){
```

```
  set_tris_e(0);
  set_tris_d(TRISD & 0xf0);
  }
char lcd_read_byte(void){
    char low,high;
    set_tris_lcd_read();
    lcd_command_port.read_write = READ1;
    delay_cycles(1);

    lcd_command_port.enable = E_UP;
    delay_cycles(1);
    high = lcd_data_port.data & 0x0f;
    lcd_command_port.enable = E_DOWN;
    delay_cycles(1);

    lcd_command_port.enable = E_UP;
    delay_us(1);
    low = lcd_data_port.data & 0x0f;
    lcd_command_port.enable = E_DOWN;
    set_tris_lcd_write();
    return( (high << 4) | low);
}
//*************** -  - ********************************************
void lcd_write_nibble(char temp_wr){
//Purpose:
    lcd_data_port.data = temp_wr;
    delay_cycles(1);
    lcd_command_port.enable = 1;
    delay_us(2);
    lcd_command_port.enable = 0;
```

```
}
//***************  ******************************************
void lcd_send_byte(char data , char data_or_command) {

     lcd_command_port.data_command = COMMAND0;
     while ( bit_test(lcd_read_byte(),7) ) ;
     lcd_command_port.data_command = data_or_command;
     delay_cycles(1);
     lcd_command_port.read_write = WRITE0;
     delay_cycles(1);
     lcd_command_port.enable = E_DOWN;
     lcd_write_nibble(data >> 4);
     lcd_write_nibble(data & 0xf);
}
//************************************************************
void lcd_gotoxy( char location, char line) {//location 0...
  char address;

  if(line!=1)
    address=40;//was80
  else
    address=0;
  address+=location;
  lcd_send_byte(0x80|address,COMMAND0);
}

void lcd_init(void){
//Purpose: initiate the lcd
  //turnning PORTA0=analog, PORTA1-7=digital
```

```
//turnning command pin of lcd to output
set_tris_lcd_write();

//delay 15msec
delay_ms(15);
lcd_command_port.enable=0;
lcd_command_port.data_command=0;
lcd_command_port.read_write=0;
lcd_data_port.data=0;
delay_ms(2);
lcd_write_nibble(0x03);//was 30  ddram address: 1:1bit,line:1bit,address:4bit
delay_ms(10);//more then 4.1msec

lcd_write_nibble(0x03);//was 30
delay_us(150);//more then 100usec

lcd_write_nibble(0x03);//was 30
delay_us(100);

lcd_write_nibble(0x02);//was 20
while(bit_test(lcd_read_byte(),7));

lcd_send_byte(0x28,COMMAND0);//was 0x28 portd[3:0]=x2 then portd[3:0]=x8
            //  4 bit low nibble, ddram address=0 first line

lcd_send_byte(0x0c,COMMAND0);//disp on

clear_lcd;
```

```
  lcd_send_byte(0x6,COMMAND0);//entry inc

  lcd_gotoxy(1,1);

}
//************************************************************
void lcd_putc( char c) {//this function works good
  switch (c) {
    case '\f'   : lcd_send_byte(1,COMMAND0);   delay_ms(2);   break;   //Clear display
    case '\n'   : lcd_gotoxy(1,2);                    break;   //Go to start of second line
    case '\b'   : lcd_send_byte(0x10,COMMAND0);             break;   //Move back one position
    default     : lcd_send_byte(c,DATA1);               break;   //send the actual character
  }
}
//*************** ******************************************
char lcd_getc( char location, char line) {
     char value;

   lcd_gotoxy(location,line);
   while ( bit_test(lcd_read_byte(),7) ); // wait until busy flag is low
   lcd_command_port.data_command=1;
   value = lcd_read_byte();
   lcd_command_port.data_command=0;
   return(value);
    return 1;
}
////////////////////////////////////////////////////////////////////////////
//********** function for keyspad and EEPROM *********************************
////////////////////////////////////////////////////////////////////////////
WRITE_LONG_EEPROM(long int n, long data)
```

```
{

  int i;
  for (i = 0;i<2;i++)
    write_eeprom(i + n,*(&data + i)) ;
}

long READ_LONG_EEPROM(long int n) {

  int i;
  long data;
  for (i = 0;i<2;i++)
    *(&data + i) = read_eeprom(i + n);
  return(data);
}



void beep()
{
  int16 p;
  for(p=0;p<=20;p++)
  {
    output_high(PIN_B3);
    delay_us(500);
    output_low(PIN_B3);
    delay_us(500);
  }//end for
}//end function
void move_digi()
{
```

```
  data[0]=data[1];
  data[1]=data[2];
  data[2]=data[3];
}//end function
void clear_data()
{
      data[0]=0;
      data[1]=0;
      data[2]=0;
      data[3]=0;

}

char get_kbd(void){
char i,k,m,j,a;
k=0xfb;
for(i=0;i<=2;i++)
  {
    m=k|0xf0;
    output_b(m);
    j=PORTB;
    j=j & 0xf0;
    delay_ms(10);
    if(j!=0xf0)
    {
        if(~keypressed)
        {
         keypressed=1;
         m = k & 0x7;
         j=j|m;
```

```
            beep();
            for(a=0;a<=11;a++)
              if(j==code[a])
              return(a);
            return(0xff);
           }//end if
      return(0xff);
      } //end if
     k=k>>1;
   } //end for
   keypressed=0;
   return(0xff);
} // end function
////////////////////////////////////////////////////////////////////////////
//============== this function for config CCP module to PWM ================
////////////////////////////////////////////////////////////////////////////
void init_motor(void)
{
   setup_ccp1(CCP_PWM_PLUS_3);
  //setup_ccp2(CCP_PWM_PLUS_3);
  // asm block same as setup_ccp2(CCP_PWM_PLUS_3) function
   #asm
      MOVLW  0xB7
      ANDWF  0xFB1,F  //T3CON not use Timer 3 in pwm2(use Timer 1)
      BCF    0xF95.4  //clear TRISD bit 4 register
      BCF    0xF8c.4  //clear LATD  bit 4 register
      MOVLW  0x3c
      MOVWF  0xFBA    //move 0x3c to ECCP1CON register(10 bit resolution)
   #endasm
   setup_timer_2(T2_DIV_BY_16,255,10);
```

```
}
////////////////////////////////////////////////////////////////////////////
//======= this function for get speed from encoder return speed(RPM) ===========
////////////////////////////////////////////////////////////////////////////
/*long gets_speed(void)
{
  long speed;
  setup_timer_1(T1_EXTERNAL_SYNC|T1_DIV_BY_1);
  set_timer1(0);
  delay_ms(100);      //change delay
  speed=(get_timer1())*10;
  return speed;
}*/
long gets_speed(void)
{
  float freq;
  long speed;
  long count =0;
  setup_timer_1 ( T1_INTERNAL | T1_DIV_BY_8 );
  while(input(PIN_C0))
  {
    delay_us(2);
    count = count+1;
    if(count>=12500)
    {
      count =0;
      speed = 0;
      goto out;
    }
  }
```

```
  while(!input(PIN_C0))
  {
   delay_us(2);
    count = count+1;
    if(count>=12500)
    {
      count =0;
      speed = 0;
      goto out;
    }
  }
  set_timer1(0);
  while(input(PIN_C0))
  {
    delay_us(2);
    count = count+1;
    if(count>=12500)
    {
      count =0;
      speed = 0;
      goto out;
    }
  }
  speed=get_timer1();
  freq =40000000/(64.0*speed);
  speed =(long)freq;
out:  return speed;
}

////////////////////////////////////////////////////////////////////////////
```

```
//========== this function for determine error of system speed ================
////////////////////////////////////////////////////////////////////////////
signed long determine_err(long nref)
{
  long n;
  signed long err;
  n=gets_speed();   // get speed motor from encoder
  printf(lcd_putc,"\f n=%4lu rpm",n);
  printf("%4lu\n\r",n);
  err=nref-n;
  //printf("\n error  = %ld\n\r",err);
  return  err;
}
//
signed int fuzzy_1(signed long err,signed long errpre )
{
  signed long errch;
  int index1,index2;
  signed int output;
  //========== lookup table for crisp output ==============================
  signed int data[19][19]={
  {-21,-21,-21,-21,-18,-14,-14,-14,-10,-7,-7,-7,-7,-7,-3,0,0,0,0},
  {-21,-21,-21,-21,-18,-14,-14,-14,-10,-7,-7,-7,-7,-7,-3,0,0,0,0},
  {-21,-21,-21,-21,-18,-14,-14,-14,-10,-7,-7,-7,-7,-7,-3,0,0,0,0},
  {-21,-21,-21,-21,-18,-14,-14,-14,-10,-7,-7,-7,-7,-7,-3,0,0,0,0},
  {-18,-18,-18,-18,-18,-14,-14,-14,-10,-7,-4,-4,-4,-4,0,3,3,3,3},
  {-14,-14,-14,-14,-14,-14,-14,-14,-10,-7,-4,0,0,0,4,7,7,7,7},
  {-14,-14,-14,-14,-14,-14,-14,-14,-10,-7,-4,0,0,0,4,7,7,7,7},
  {-11,-11,-11,-11,-11,-11,-11,-11,-7,-4,-1,3,3,3,4,7,7,7,7},
  {-10,-10,-10,-10,-10,-10,-10,-10,-6,-3,0,4,4,4,4,7,7,7,7},
```

```
  {-7,-7,-7,-7,-7,-7,-7,-7,-3,0,3,7,7,7,7,7,7,7,7},
  {-7,-7,-7,-7,-4,-4,-4,-4,0,3,3,7,7,7,10,10,10,10,10},
  {-7,-7,-7,-7,-4,-3,-3,-3,1,4,4,7,7,7,11,11,11,11,11},
  {-7,-7,-7,-7,-4,0,0,0,4,7,7,7,7,7,11,14,14,14,14},
  {-7,-7,-7,-7,-4,0,0,0,4,7,7,7,7,7,11,14,14,14,14},
  {-3,-3,-3,-3,0,4,4,4,4,7,10,11,11,11,15,18,18,18,18},
  {0,0,0,0,3,7,7,7,7,7,10,14,14,14,18,21,21,21,21},
  {0,0,0,0,3,7,7,7,7,7,10,14,14,14,18,21,21,21,21},
  {0,0,0,0,3,7,7,7,7,7,10,14,14,14,18,21,21,21,21},
  {0,0,0,0,3,7,7,7,7,7,10,14,14,14,18,21,21,21,21}
};
//=================================================================
=========

      errch=err-errpre;
      // quantize error for discreate value

      ///
      if(err>=340)                          err=9;
      else if((err>=300)&&(err<340))        err=8;
      else if((err>=260)&&(err<300))        err=7;
      else if((err>=220)&&(err<260))        err=6;
      else if((err>=180)&&(err<220))         err=5;
      else if((err>=140)&&(err<180))         err=4;
      else if((err>=100)&&(err<140))        err=3;
      else if((err>=60)&&(err<100))         err=2;
      else if((err>=20)&&(err<60))          err=1;
      else if((err>=-20)&&(err<20))         err=0;
      else if((err>=-60)&&(err<-20))        err=-1;
      else if((err>=-100)&&(err<-60))        err=-2;
```

```
  else if((err>=-140)&&(err<-100))          err=-3;
  else if((err>=-180)&&(err<-140))          err=-4;
  else if((err>=-220)&&(err<-180))          err=-5;
  else if((err>=-260)&&(err<-220))          err=-6;
  else if((err>=-300)&&(err<-260))          err=-7;
  else if((err>=-340)&&(err<-300))          err=-8;
  else                                      err=-9;
  //quantize change of error for discreate value
  if(errch>=180)                            errch=9;
  else if((errch>=160)&&(errch<180))        errch=8;
  else if((errch>=140)&&(errch<160))        errch=7;
  else if((errch>=120)&&(errch<140))        errch=6;
  else if((errch>=100)&&(errch<120))        errch=5;
  else if((errch>=80)&&(errch<100))         errch=4;
  else if((errch>=60)&&(errch<80))          errch=3;
  else if((errch>=40)&&(errch<60))          errch=2;
  else if((errch>=20)&&(errch<40))          errch=1;
  else if((errch>=-20)&&(errch<20))         errch=0;
  else if((errch>=-40)&&(errch<-20))        errch=-1;
  else if((errch>=-60)&&(errch<-40))        errch=-2;
  else if((errch>=-80)&&(errch<-60))        errch=-3;
  else if((errch>=-100)&&(errch<-80))       errch=-4;
  else if((errch>=-120)&&(errch<-100))      errch=-5;
  else if((errch>=-140)&&(errch<-120))      errch=-6;
  else if((errch>=-160)&&(errch<-140))      errch=-7;
  else if((errch>=-180)&&(errch<-160))      errch=-8;
  else                                      errch=-9;
index1=err+9;
index2=errch+9;
output=data[index1][index2];  //lookup table
```

```
    return output;

} // end fuzzy_1 function
//
void set_speed_m1(char s)
{
  set_pwm1_duty(s);
}
//
void set_speed_m2(char s)
{
  set_pwm2_duty(s);
}
//
```