

PONTIKAKIS ANTHONY
NIK. ANASTASAKI 3
CHANIA CRETA
T/K 73100
GREECE

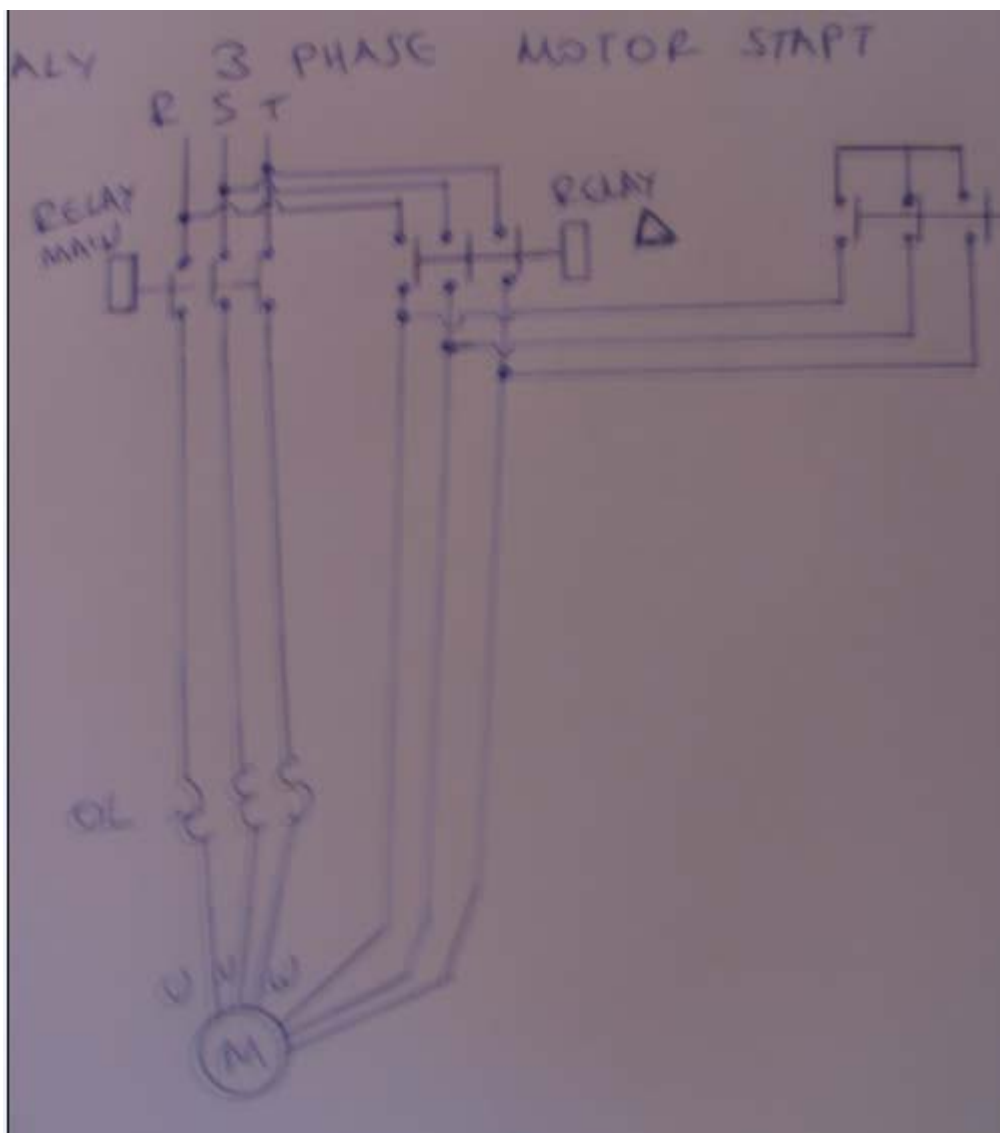
Email: pontikakis2@gmail.com

DEAR SIR

I DO NOT KNOW IF THE COMPETITION FOR THE R8C IS STILL IN EFFECT.

IN ANY CASE I MADE THIS PROGRAM FOR SMOOTH START

A 3 FASE MOTOR WHERE THIS PUMP WORKS FOR the PUMPING of WATER AND TRANSPORT of THIS IN ALTITUDE 80M.



THIS IS A NORMALLY START CIRCUIT OF A 3 PHASE MOTOR

1ST CLOSE THE MAIN RELAY AND THE Y RELAY AND AFTER FEW SECONDS THE Y RELAY OPEN AND THE Δ RELAY CLOSE.

IS NAMED SOFT START AND SUBSTANTIALLY ALTERS THE DUTY CYCLE IN THE SCR OF BENEFIT.

O TIMER Z GIVES 2 KHz OUTPUT PROGRAMMABLE DUTY CYCLE
AND the PUMP BEGINS SMOOTHLY WITH 10%, 20%, 40%, 60%, 80%, 100% o DC
P3_0=1 (A RELAY BYPASS THE SCR)
P3_1=0 (TIMER Z STOPS COUNTING)

WHEN THE PUMP STOPS WE WANT SMOOTH RESERVATION of THIS (REASON of PRESSURE THAT IT HAS WATER IN THE ALTITUDE OF + 80 METERS). THUS the RESERVATION of THIS BECOMES WITH BACKWARD WAY

- a) TIMER Z START COUNTING
- b) P3_1 ENERGIZED
- c) P3_0=0 (BYPASS RELAY OFF)
- d) 100%, 80%, 60%, 40%, 20%, 10% DC
- e) P3_1=0 TIMER Z STOP COUNTING <>

FOR THE PROTECTION OF PUMP FROM THE CASE:

- a) OVERLOADED
- B) UNLOADED THE CASE WHERE THE PUMP WORKS WITHOUT CHARGE

WE USE 3 TRANSFORMERS ONE FOR EACH FACE (A, B, C)
THAT INDICATE THE WORKING AMP OF THE MOTOR.
EACH OF THE ABOVE CONDITIONS IS EMERGENCY AND THE MOTOR MUST STOPPED.

TO LCD THAT I USE IS 2X16 CHAR. AND I CANNOT PRESENT A LOT OF THINGS IN THIS.
(FOR THIS REASON I ORDERED 4X20 CHAR.)
I HAVEN'T RECEIVED YET.

STILL I MADE A MENU OF CHOICE OF OPERATION WHERE THE USER CAN SELECT THE INTERMEDIATE TIMES
AT SOFT_START OR SOFT_STOP (FAST OR SLOW IN 4 MODES).

STILL THEN WE CAN SELECT THE OPERATION MOTOR (MOT1, OR OR MOT2 OR MOT3 OR MOT4)

FOR THIS REASON I USE TWO DECODER:

1ST DECODER
P1_0 A0
P1_1 A1
P3_1 INPUT OF DECODER

OUTPUTS

1ST MOTOR 1 SCR
2ND MOTOR 2 SCR
3RD MOTOR 3 SCR
4TH MOTOR 4 SCR

2ND DECODER
INPUT OF THE DECODER ALWAYS 1 (HIGH)

P1_4 A0
P1_5 A1
OUTPUTS

1ST BYPASS RELAY MOT1
2ND BYPASS RELAY MOT2
3RD BYPASS RELAY MOT3
4TH BYPASS RELAY MOT4

USUALLY THE PUMP SYSTEM HAVE 2 OR 3 PUMPS ALWAYS ONE PUMP WORKING, THE OTHER THEY IS BACKUP.

I USE THE INT1 INTERRUPT (P1_7 PIN) FOR ENTRANCE TO MENU
AND THE P1_6 AS AN INPUT SWITCH SELECTION
AND THE P4_5 AS AN ENTER INPUT

IN THESE SUBROUTINES i USE THE ON CHIP OSC (8Mhz) JUST TO
TEST IT.

THERE ARE SOME SPACE PROBLEMS WITH THE LCD (IT'S ONLY
2X16 CHAR.)

THESE PROGRAM I TEST IT ONLY ON EV. BOARD AND I NEED TO
ADD AN I/O UNIT TO TEST IT IN A 3 PHASE MOTOR.

THERE ARE 3 VOLTMETERS AND MAX AMP RATING IS 250A,
IN THE EV. BOARD i ADD 3 POT 10K FOR THE 3 ANA LOGS INPUTS
I USE THE AN (6) P0_1, AN (10) P1_2, AN (11) P1_3.

IN a REAL CONDITION i USE TRANSFORMERS 10:1 ONE FOR FASE
OF THE MOTOR THEN a RECTIFIER AND A VOLTAGE DIVIDER TO GET THE RIGHT INPUT FOR
THE A2D.

THE PROGRAM:

```

/*****
/*
/*
/* FILE      :timerZ_output.c
/* DATE      :Thu, Jul 06, 2006
/* DESCRIPTION :Main Program
/* CPU TYPE   :R8C/13
/* AUTHOR PONTIKAKIS ANTHONY
/* This file is generated by Renesas Project Generator (Ver.4.0).
/*
*****/

```

```

#include "sfr_r813.h"
#pragma interrupt int1;
long t;
long zeit;
int i;
unsigned int counter;
long counter1;
long counter2;
unsigned int counter3;
static unsigned char Port0;
void lcddata(unsigned char data);
void delayus (unsigned int micros);
void lcd_text (char text[20]);
void lcd_pos (unsigned int Zeile, unsigned int Spalte);
void lcdctrl(unsigned char data);
void uart0_init(void);
void sendTxd0(unsigned char data);
unsigned char receiveRxd0 (void);
unsigned int ad_in(unsigned char ch);
unsigned int ad_in1(unsigned char ch);
void voltometer();
void routine();
void routine1();
void count3();
void start();
void start1();
void start2();
void start3();
void start4();
void emergency_stop();
void stop80();
void stop60();
void stop40();
void voltometer1();
void voltometer2();

```

```

void routine()
{
    lcdctrl(0X01); //clear Lcd
    delayus(100000);
        prc0=1; //protect disable
        ocd0=0;
        ocd1=0;
        ocd2=1; // on chip oscilator
    hr00=1; //high speed on
        hr01=1; // high speed oscilator selected
        cm06=0;
        cm14=0; //low speed on
        cm16=1;

```

```

        cm17=1; //division by 16 mode
        cm05=1; //main clock of
        asm("nop");
        asm("nop");
asm("nop");
        asm("nop"); //wai for stability
        prc0=0; //protect enable
        while(oed3=0) {} //wait to stop the main clock
        asm("nop");
        if(counter==0){
            lcd_pos(1,1);
            lcd_text(" ");
            lcd_pos(1,11);
            lcd_text(" ");
            lcd_pos(2,1);
            lcd_text(" ");
            lcd_pos(2,11);
            lcd_text(" ");}
        else if(counter==1)
            {lcd_pos(1,1);
             lcddata(0Xff);}
        else if(counter==2)
            {lcd_pos(1,11);
             lcddata(0Xff);}
        else if(counter==3)
            {lcd_pos(2,1);
             lcddata(0Xff);}
        else if(counter==4)
            {lcd_pos(2,11);
             lcddata(0Xff);}
        lcd_pos(1,2);
        lcd_text("PROG1");
        lcd_pos(1,12);
        lcd_text("PROG2");
        lcd_pos(2,2);
        lcd_text("PROG3");
        lcd_pos(2,12);
        lcd_text("PROG4");
        delayus(100);
while(p4_5==1){

        if(p1_6==0){
            while(p1_6==0){} //wait to be "1"
            if(counter>4){
                counter=1;
            }
        }
        else {counter++;}
            if(counter==1){
                counter1=1000000;
                counter2= 800000;

```

```

        lcd_pos(2,11);
        lcd_text(" ");
        lcd_pos(1,1);
        lcddata(0Xff);
    }

    else if(counter==2){
        counter1=2000000;
        counter2=1000000;
        lcd_pos(1,1);
        lcd_text(" ");
        lcd_pos(1,11);
        lcddata(0Xff);
    }

    else if(counter==3){
        counter1=3000000;
        counter2=2500000;
        lcd_pos(1,11);
        lcd_text(" ");
        lcd_pos(2,1);
        lcddata(0Xff);
    }

    else if(counter==4){
        counter1=4000000;
        counter2=3000000;
        lcd_pos(2,1);
        lcd_text(" ");
        lcd_pos(2,11);
        lcddata(0Xff);
    }
}
}

```

```

void routine1(){
    lcdctrl(0x01); //clear lcd
    delayus(500);

    if(counter3==0){
        lcd_pos(1,1);
        lcd_text(" ");
        lcd_pos(1,11);
        lcd_text(" ");
        lcd_pos(2,1);
        lcd_text(" ");
        lcd_pos(2,11);
        lcd_text(" ");}
    else if(counter3==1)
        {lcd_pos(1,1);
        lcddata(0Xff);}
    else if(counter3==2)

```

```

        {lcd_pos(1,11);
          lcddata(0Xff);}
else if(counter3==3)
    {lcd_pos(2,1);
      lcddata(0Xff);}
else if(counter3==4)
    {lcd_pos(2,11);
      lcddata(0Xff);}
      asm("nop");
  lcd_pos(1,2);
  lcd_text("MOTO1");
  lcd_pos(1,12);
  lcd_text("MOTO2");
  lcd_pos(2,2);
  lcd_text("MOTO3");
  lcd_pos(2,12);
  lcd_text("MOTO4");
  delayus(100);
while(p4_5==1){
    if(p1_6==0){
        while(p1_6==0){} //wait to be "1"
        if(counter3>4){
            counter3=1;
            p1_0=0;

            p1_1=0;
            p1_4=0;
            p1_5=0;

            lcd_pos(2,11);
            lcd_text(" ");
            lcd_pos(1,1);
            lcddata(0Xff);
        }
    }
    else {counter3++;}
        if(counter3==1){
            p1_0=0;

            p1_1=0;
            p1_4=0;
            p1_5=0;

            lcd_pos(2,11);
            lcd_text(" ");
            lcd_pos(1,1);
            lcddata(0Xff);
        }
    else if(counter3==2){
        p1_0=1;

        p1_1=0;
        p1_4=1;
        p1_5=0;

        lcd_pos(1,1);

```

```

        lcd_pos(1,11);
        lcd_text(" ");
        lcddata(0Xff);
    }
    else if(counter3==3){
        p1_0=0;
        p1_1=1;
        p1_4=0;
        p1_5=1;
        lcd_pos(1,11);
        lcd_text(" ");
        lcd_pos(2,1);
        lcddata(0Xff);
    }
    else if(counter3==4){
        p1_0=1;
        p1_1=1;
        p1_4=1;
        p1_5=1;
        lcd_pos(2,1);
        lcd_text(" ");
        lcd_pos(2,11);
        lcddata(0Xff);
    }
}
}
void count3(){
    if (counter3==0)
        {} // NO OPERATION
    else if (counter3==1){
        p1_0=0;
        p1_1=0;
        p1_4=0;
        p1_5=0;
    }
    else if(counter3==2){
        p1_0=1;
        p1_1=0;
        p1_4=1;
        p1_5=0;
    }
    else if(counter3==3){
        p1_0=0;
        p1_1=1;
        p1_4=0;
        p1_5=1;
    }
    else if(counter3==4){
        p1_0=1;

```



```

        }
    }
    void start()
    {count3();
    tzocnt=0;
    tzmod0=1; //programable output mode
    tzmod1=0;
    tzwc=1;
    tzopl=0;
    prez=250-1;
    tzpr=4-1;
    tzsc=36-1;
    tzs=1; //start counting
    }
    void start1()
    {count3();
    tzocnt=0;
    tzck0=0; //count source f1
    tzck1=0;
    tzmod0=1; //programable output
    tzmod1=0;
    tzwc=1;
    tzopl=0;
    prez=250-1;
    tzsc=32-1;
    tzpr=8-1;
    tzs=1; //start counting

    }
    void start2()
    {count3();
    tzocnt=0;
    tzck0=0; //count source f1
    tzck1=0;
    tzmod0=1; //programable output
    tzmod1=0;
    tzwc=1;
    tzopl=0;
    prez=250-1;
    tzsc=24-1;
    tzpr=16-1;
    tzs=1; //start counting

    }
    void start3()
    {count3();

```

```

p1_1=1;
p1_4=1;
p1_5=1;

```

```

tzck0=0; //count source fl
tzck1=0;
tzmod0=1; //programable outpout
tzmod1=0;
tzwc=1;
tzopl=0;
prez=250-1;
tzsc=16-1;
tzpr=24-1;
tzs=1; //start counting

```

```

}
void start4()
{count3();
tzocnt=0;
tzck0=0; //count source fl
tzck1=0;
tzmod0=1; //programable outpout
tzmod1=0;
tzwc=1;
tzopl=0;
prez=250-1;
tzsc=8-1;
tzpr=32-1;
tzs=1; //start counting

```

```

}
void emergency_stop(){
start4();
lcd_pos(1,1);
lcd_text("80%      ");
lcd_pos(2,1);
lcd_text("DUTTY    CYCLE");
for (t=0; t<500000; t++);
start3();
lcd_pos(1,1);
lcd_text("60%      ");
lcd_pos(2,1);
lcd_text("DUTTY    CYCLE");
for (t=0; t<500000; t++);
start2();
lcd_pos(1,1);
lcd_text("40%      ");
for (t=0; t<500000; t++);
start1();
lcd_pos(1,1);
lcd_text("20%      ");
for (t=0; t<500000; t++);
p3_0=0;
tzocnt=1;

```

```

p3_1=0;
}
void stop80(){
start3();
lcd_pos(1,1);
lcd_text("60%      ");
lcd_pos(2,1);
lcd_text("DUTTY    CYCLE");
for (t=0; t<500000; t++);
start2();
lcd_pos(1,1);
lcd_text("40%      ");
for (t=0; t<500000; t++);
start1();
lcd_pos(1,1);
lcd_text("20%      ");
for (t=0; t<500000; t++);
p3_0=0;
tzocnt=1;
p3_1=0;
}
void stop60(){
start2();
lcd_pos(1,1);
lcd_text("40%      ");
lcd_pos(2,1);
lcd_text("DUTTY    CYCLE");
for (t=0; t<500000; t++);
start1();
lcd_pos(1,1);
lcd_text("20%      ");
for (t=0; t<500000; t++);
p3_0=0;
tzocnt=1;
p3_1=0;
}
void stop40(){
start1();
lcd_pos(1,1);
lcd_text("20%      ");
lcd_pos(2,1);
lcd_text("DUTTY    CYCLE");
for (t=0; t<500000; t++);
p3_0=0;
tzocnt=1;
p3_1=0;
}
static unsigned char Port0;
void delayus (unsigned int micros)
{

```

```

        unsigned long t;
        for (t = 0; t < micros; t++)
        {
            asm("nop");
        }
    }
    void lcddata(unsigned char data)
    {
        delayus(100);
        Port0 = data & 0xF0;
        Port0 = Port0 + 0x04;          //rs = 1
        p0 = Port0;
        p0 = Port0 + 0x08;             //E = 1
        asm("nop");
        p0 = Port0;                    //E = 0
        Port0 = ((data & 0x0f)<<4);
        Port0 = Port0 + 0x04; //rs = 1
        p0 = Port0;
        p0 = Port0 + 0x08;             //E = 1
        asm("nop");
        p0 = Port0;                    //E = 0
        delayus (100);
    }

```

```

    void lcdctrl(unsigned char data)
    {
        Port0 = data & 0xF0;
        p0 = Port0;
        p0 = Port0 + 0x08;             //E = 1
        asm("nop");
        p0 = Port0;                    //E = 0
        Port0 = ((data & 0x0f)<<4);
        Port0 = Port0 + 0x01;
        p0 = Port0;
        p0 = Port0 + 0x08;             //E = 1
        asm("nop");
        p0 = Port0;                    //E = 0
        delayus (100);
    }

```

```

    void initlcd (void)
    {
        delayus(15000);
        lcdctrl(0x28);
        delayus(5000);
        lcdctrl(0x28);
        delayus(1000);
        lcdctrl(0x28);
        delayus(1000);
        lcdctrl(0x0c);
    }

```

```

        delayus(1000);
        lcdctrl(0x01);
        delayus(5000);
    }
    void uart0_init(void){
        pd1_4 = 1;    // TxD0 port direction = output
        pd1_5 = 0;    // RxD0 port direction = input
        u0mr = 0x05;   // UART0 transmit/receive mode
        u0c0 = 0x00;   // UART0 transmit/receive control
        u0rrm = 0;     // Continuous receive mode disabled
        u0brg = 130-1; // 9600 baud @20MHz
        re_u0c1 = 1;   // Reception enabled
    }

    void sendTxd0(unsigned char data)
    {
        while (ti_u0c1 == 0); //Wait for transmission buffer empty
        u0tbl = data;         // Set transmission data
        te_u0c1 = 1;         // Transmission enabled
    }

    unsigned char receiveRxd0 (void)
    {
        unsigned char data;
        unsigned char dummy;
        while (ir_s0ric == 0); //Wait for received data
        ir_s0ric = 0;          //Clear serial reception flag
        data = u0rbl;          // Get reception data
        dummy = u0rbh;         // Get error
        re_u0c1 = 1;          // Reception enabled
        return data;
    }
    void lcd_text (char text[20])
    {
        unsigned int s;
        s = 0;
        while ((!(text[s] == 0)) & (s < 20))
        {
            lcddata (text[s]);
            s = s + 1;
        }
    }

    void lcd_pos (unsigned int Zeile, unsigned int Spalte)
    {
        lcdctrl (0x80 + Spalte-1 + 0x40*(Zeile-1));
        delayus(100);
    }

```

```

unsigned int ad_in(unsigned char ch)
{
    adcon0 = 0x80 + ch;    //Port P0 group
    adcon1 = 0x28;         //10-bit mode
    adst = 1;              //Conversion start
    while(adst == 1){}     //Wait A/D conversion
    return ad;              //AD value
}

```

```

unsigned int ad_in1(unsigned char ch)
{
    adcon0 = 0x90 + ch;    //Port P1 group
    adcon1 = 0x28;         //10-bit mode
    adst = 1;              //Conversion start
    while(adst == 1){}     //Wait A/D conversion
    return ad;              //AD value
}

```

```

void voltometer()
{
    float u;
    unsigned char c;
    u = (float) ad_in(6);
    u = (u / 1023.0 * 5.0)/2; //max value (timi) 250.0
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (0x2e); //koma dhl. teleia
    lcddata (c+48);
    lcddata(65); //A ampere
    delayus(100);
}

```

```

void voltometer1()
{
    float u;
    unsigned char c;
    u = (float) ad_in1(6);
}

```

```

    u = (u / 1023.0 * 5.0)/2; //max value (timi) 250.0
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (0x2e); //koma dhl. teleia
    lcddata (c+48);
    lcddata(65);
    delayus(100);
}

void voltometer2()
{
    float u;
    unsigned char c;
    u = (float) ad_inl(7);
    u = (u / 1023.0 * 5.0)/2; //max value (timi) 250.0
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (c+48);
    u = u - c;
    u = u * 10;
    c = u;
    lcddata (0x2e); //koma dhl. teleia
    lcddata (c+48);
    lcddata(65); //A AMPERE
    delayus(100);
}

void main(void)
{
/*-----*/
- Change on-chip oscillator clock to Main clock -
/*-----*/

prc0 = 1;          /* Protect off */

```

```

cm13 = 1;          /* Xin Xout */
cm15 = 1;          /* XCIN-XCOUT drive capacity select bit : HIGH */
cm05 = 0;          /* Xin on */
cm16 = 0;          /* Main clock = No division mode */
cm17 = 0;
cm06 = 0;          /* CM16 and CM17 enable */
asm("nop");        /* Waiting for stable of oscillation */
asm("nop");
asm("nop");
asm("nop");
ocd2 = 0;          /* Main clock change */
prc0 = 0;          /* Protect on */
pd4_5=0;
    pd1_0=1; //output
    pd1_1=1; //output
    pd1_2=0; //input
    pd1_3=0; //input
    pd3_2=0; //input
    pd3_1=1; //output
    pd3_0=1; //output
    pd3_3=0; //input
    pd1_4=1; //UART0 TX OUTPUT
    pd1_5=1; // UART0 RX OUTPUT
    pd1_6=0; //input
    pd1_7=0; //input interrupt INT1
    pd4_5=0; //input
    Port0 = 0xff;
    p0 = 0xfd;      //AN6 INPUT ONLY
    prc2 = 1;        //protect off
pd0=0xfd; //AN6 INPUT ONLY
    prc2=0; //protect on
txck0=0;
txck1=0; //f1 selected
    txmod0=0;
    txmod1=0; //timer mode
ir_txic=0; //flag clear
prex=160-1;
tx=250-1; //20mSec counter
txs=0; //stop counting
    int1ic=0X05; //priority level
r0edg=1; // falling edge
    initlcd(); //intialize LCD
asm( "\tFSET I"); /* Enable interrupt */
counter=0;
counter1=0;
counter2=0;
counter3=0;
tzs=0; //stop counting counting
int1ic=0X05; //priority level
while(counter==0 && counter3==0)

```



```

{lcd_pos(1,1);
  lcd_text("SELECT PROG MODE");
  lcd_pos(2,1);
  lcd_text("PRESS MOD SWITCH");
}
while (1) {
  while(p3_2==1) //start button not pressed
    {count3();
      tyocnt=1;
      p3_1=0;

      lcd_pos(1,1);

      lcd_text("MOT. STOP ");
      lcd_pos(1,11);
      voltmeter();
      lcd_pos(2,1);
      voltmeter1();
      lcd_pos(2,7);
      lcd_text(" ");
      lcd_pos(2,11);
      voltmeter2();
    }
    asm("nop");
    int1ic=0X00; //priority level 0 interrupt deenergized
    count3();
    start();

    lcd_pos(1,1);

    lcd_text("10%      ");
    lcd_pos(2,1);
    lcd_text("DUTTY    CYCLE");
    for(t=0; t<counter1; t++);
    start1();
    if(p3_3==1)
    {lcd_pos(1,1);
      lcd_text("20%      ");

      lcd_pos(2,1);

      lcd_text("DUTTY    CYCLE");}
    else
    {tzocnt=1;
      p3_1=0;
      p3_0=0;
      goto text;}
    delayus(100);
    if(ad_in(6)<=450 && ad_in1(6)<=450 && ad_in1(7)<=450){}
    else
    {while(1)
      {lcd_pos(1,1);
        lcd_text("MOTOR OVERLOADED");
        lcd_pos(2,1);
        lcd_text(" EMERGENCY STOP ");
        p3_0=0;

```

```

tzoCnt=1;
p3_1=0;}
}

for(t=0; t<counter1; t++);
start2();
if(p3_3==1)

    {lcd_pos(1,1);
    lcd_text("40%      ");

lcd_pos(2,1);

    lcd_text("DUTTY    CYCLE");}
else
{stop40();
goto text;}
delayus(100);
if(ad_in(6)<=500 && ad_in1(6)<=500 && ad_in1(7)<=500){}
else
    {stop40();
    while(1)
        {lcd_pos(1,1);
        lcd_text("MOTOR OVERLOADED");
        lcd_pos(2,1);
        lcd_text(" EMERGENCY STOP ");
        p3_0=0;}
    }

for(t=0; t<counter1; t++);
start3();
if(p3_3==1)

    {lcd_pos(1,1);
    lcd_text("60%      ");

lcd_pos(2,1);

    lcd_text("DUTTY    CYCLE");}
else
{stop60();
goto text;}
delayus(100);
if(ad_in(6)<=550 && ad_in1(6)<=550 && ad_in1(7)<=550){}
else
    {stop60();
    while(1)
        {lcd_pos(1,1);
        lcd_text("MOTOR OVERLOADED");
        lcd_pos(2,1);
        lcd_text(" EMERGENCY STOP ");
        p3_0=0;}
    }

for(t=0; t<counter1; t++);
start4();
if(p3_3==1)

    {lcd_pos(1,1);
    lcd_text("80%      ");

```

```

        lcd_pos(2,1);

        lcd_text("DUTTY    CYCLE");}
    else
    {stop80();
    goto text;}
    delayus(100);
    if(ad_in(6)<=600 && ad_in1(6)<=600 &&
ad_in1(7)<=600){count3();}

    else
    {stop80();
    while(1)
        {lcd_pos(1,1);
        lcd_text("MOTOR OVERLOADED");
        lcd_pos(2,1);
        lcd_text(" EMERGENCY STOP ");
        p3_0=0;}
    }

    for(t=0; t<counter1; t++);
        count3();
    tzocnt=1;

    p3_1=1;
    p3_0=1;

    lcd_pos(1,1);
        lcd_text("100%    ");

    lcd_pos(2,1);
        lcd_text("DUTTY    CYCLE");
    for(t=0; t<counter1; t++);
        if(p3_3==1)
    {count3();

    tzs=0; //stop counting
    p3_1=0;
    p3_1=1;
    lcd_pos(1,1);
    lcd_text("MAIN REL  ");}
    else
    {stop80();
    goto text;}

    asm("nop");

    if(ad_in(6)<=650 && ad_in1(6)<=650 && ad_in1(7)<=650)
        {count3();
        tzopl=1;
        tzs=0; //stop counting
        p3_1=0;
        p3_0=1;}
    else{emergency_stop();
        while(1)
            {lcd_pos(1,1);
            lcd_text("MOTOR OVERLOADED");
            lcd_pos(2,1);
            lcd_text(" EMERGENCY STOP ");

```

```

        p3_0=0;}
    }

    while(1){
        if(p3_3==1 && (ad_in(6)<=653 || ad_in1(6)<=653 ||
ad_in1(7)<=653)) //stop button not pressed
        {if(p3_3==1 && (ad_in(6)<=100 || ad_in1(6)<=100 || ad_in1(7)<=100))
            {emergency_stop();

                                asm("nop");
                                while(1){lcd_pos(1,1);
                                lcd_text("NO LOAD TO
MOTOR");

                                lcd_pos(2,1);
                                lcd_text(" EMERGENCY STOP
");

                                p3_0=0;
                                tzocnt=1;
                                p3_1=0;
                                tzs=0;}
                                }
                                else{
                                tzocnt=0;
                                lcd_pos(1,11);
                                voltometer();
                                lcd_pos(2,1);
                                voltometer1();
                                lcd_pos(2,11);
                                voltometer2();}
                                }
        else if(p3_3==1 && (ad_in(6)>=654 || ad_in1(6)>=654 || ad_in1(7)>=654))
            { emergency_stop();
                                asm("nop");
                                while(1){lcd_pos(1,1);
                                lcd_text("MOTOR
OVERLOADED");

                                lcd_pos(2,1);
                                lcd_text(" EMERGENCY STOP
");

                                p3_0=0;
                                tzocnt=1;
                                p3_1=0;}
                                }

        else if(p3_3==0 && ad_in(6)<=653 &&
ad_in1(6)<=653 && ad_in1(7)<=653)
        {
            asm("nop");
            count3();
            tzocnt=0;

```

```

start4();
p3_0=0;
lcd_pos(1,1);
lcd_text("80%      ");
lcd_pos(2,1);
lcd_text("DUTTY    CYCLE");
for(t=0; t<counter2; t++);

start3();

lcd_pos(1,1);
lcd_text("60%      ");

lcd_text("DUTTY    CYCLE");
for(t=0; t<counter1; t++);

start2();

lcd_pos(1,1);
lcd_text("40%      ");

lcd_text("DUTTY    CYCLE");
for(t=0; t<counter1; t++);

start1();

lcd_pos(1,1);
lcd_text("20%      ");

lcd_text("DUTTY    CYCLE");
for(t=0; t<counter1; t++);
tzocnt=1;

lcd_pos(1,1);

lcd_pos(1,11);

lcd_pos(2,1);

lcd_pos(2,11);

voltmeter();

voltmeter1();

voltmeter2();

p3_1=0;
p3_0=0;
count3();
goto text;}

}

text: int1ic=0X05; //priority level energized

}

}
/*****INT1 INTERRUPT*****/
void int1()
{
    routine ();
    delayus(500);
    routine1();
    prc0 = 1;      /* Protect off */

```

```

cm13 = 1;          /* Xin Xout */
cm15 = 1;          /* XCIN-XCOUT drive capacity select bit : HIGH */
cm05 = 0;          /* Xin on */
cm16 = 0;          /* Main clock = No division mode */
cm17 = 0;
cm06 = 0;          /* CM16 and CM17 enable */
asm("nop");        /* Waiting for stable of oscillation */
asm("nop");
asm("nop");
asm("nop");
ocd2 = 0;          /* Main clock change */
prc0 = 0;          /* Protect on */
lcdctrl(0X01);
delayus(500);
}

```

I HAVE A SMALL PROBLEM WITH THE 3 FACE AMPS IT HAVE TO ALL SET TO THE VALUE MORE THAN 200A TO EMERGENCY STOP THE MOTOR.
THIS IS NOT A SIRIUS PROBLEM AND CAN BE EASY FIXED.

ALSO I IMPROVE THE PROGRAM BY ADDING A REAL TIME CLOCK TO MEASURE THE WORKINGS OF THE MOTOR SO I CAN CHANGE MOTOR EVERY (FOR EXAMPLE) 4 HOURS OR IF THE USER WANTS SELECT THE MOTOR MANUAL.

ALSO I NEED TO ADD ONE MORE INPUT (REMEMBER THAT I USE ALL THE INPUTS AND PROBABLY I'LL USE THE P0_0 (TXD1) OR P3_7 (RXD1) FOR AN INPUT FOR A PRESSURE SWITCH TO SOFT STOP THE MOTOR IF THE WATER PRESSURE REACH A HIGH LEVEL AND SOFT START THE MOTOR IF THE PRESSURE DROPS BELOW A CERTAIN VALUE.

FOR THE REAL TIME CLOCK I WILL USE THE TIMER X AND THE PROGRAM WILL LOOK LIKE BELOW.

DECLARATION

```

#pragma interrupt timer_x;
unsigned long counter;
unsigned int seconds;
unsigned int minutes;
unsigned int hours;
unsigned int days;
unsigned int dayss;
unsigned int dayss1;
unsigned int days1;

```

MAIN FUNCTION

```

/*****initialise timer x*****/

```

```

txck0=0;
txck1=1; //f32 epilogh
prex=250-1;
tx=250-1; //(20Mz/32)(1/250)(1/250)=0.1sec or 10Hz
txic=0X06; // PRIORITY LEVEL
asm("nop");
/*****

```

```

*****/
*****/zero clock*****/
dayss=0;
counter=0;
seconds=0;
minutes=0;
hours=0;
days=0;
txs=1; //START COUNTING
/*****

```

THE TIMER OVERFLOW EVERY 0.1SEC OR 10Hz
 THE DAY THAT MEASURE 7 (ONE WEEK) THAT COULD BE CHANGED EASILY IF WE WANT
 THE TIMER TO MEASURE 30 DAYS OR SOMETHING ELSE.

```

/*****INTERRUPT ROUTINE*****/
void timer_x(void){

if(counter>=10){
counter=0;
seconds++;}
else counter++;

if(seconds>=60){
seconds=0;
minutes++;}

if(minutes>=60){
minutes=0;

```

```
        hours++;}

if(hours>=60){
    hours=0;
    days++;
}

if(days>=24){
    days=0;
    dayss++;
}
if(dayss>=8){
    dayss=0;}
}
```

/******

THANKS A LOT
PONTIKAKIS ANTHONY