```
/*  Yeasu G-800 rotor control - target is MC-NOVE
    2 loops: Fast and Slow: KJ1K/*
    Copyright 2010-2013 Sigurd Kimpel
    This program is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License

          http://creativecommons.org/licenses/by-nc-sa/3.0/

          http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode

    2010-11-14 original code - poor performance
    2012-12-30 updated code, fixed hardware problems - better performance
    2013-01-01 fixed null limit check coding error and tweeked map
               function to minimize scale factor error.
    2013-01-24 integrated trim procedure into main loop and installed look up table.
    */
#include <avr/pgmspace.h>

#define  TRUE  1
#define  FALSE  0

// SPI defines
#define DATAOUT 11 //MOSI
#define DATAIN 12 //MISO - part of builtin SPI, not used for this application
#define SPICLOCK 13 //sck
#define SLAVESELECT 10 //ss

// other pins
#define analogInPin A0  // Analog input pin from the potentiometer
#define lightSensor A5  // Photocell input
#define PolarityPin 4  // H switch control (DIRA)
//#define Fast_pin 9
//#define Slow_pin 8
#define CW_pin  7
#define CCW_pin 6
#define PWMA_pin 3

// Algorithm variables
int aAng = 180;
int cAng = 180;
int ang_error = 0;
int null;      // when angle error is less than null you are done
int narrow=1;  // narrow null limit in deg
int wide=5;    // wide null  limit in deg
char cmd[4] = {'c', ' ', ' ', ' '};
const char LF= 0x0a;
const char CR= 0x0d;
int sf[4] = {0, 100, 10, 1};
int index;  // number of characters rx on serial interface
const int Kpos = 100;  // output value bits per degree error (position gain)
int CW_state = 0;  // not active
int CCW_state = 0;  // not active
int state=1;      // define 8 states, State 0 used for "slow" tasks
int display_busy = FALSE;

// Analog variables
int sensorValue = 0;        // value read from the pot
double calcOutput = 0;     // Kpos* error =  pwma value
int cmdOutput;             // int version of calcOutput limited to 255
int polarity;               // HIGH for CW, LOW for CCW
int ambientLight;            // measured value of ambient light
char bright=0x0f;
char dim=0x3f;
int bump=3;                  // change cAng by "bump" degrees
// Look up table to convert bits to angle follows;
PROGMEM  prog_uint16_t a2d10[1024]= {-20,-20,-19,-19,-18,-18,-18,-17,-17,-16,-16,-15,
                  -15,-15,-14,-14,-13,-13,-13,-12,-12,-11,-11,-10,
```

```
67                  -10,-10,-9,-9,-8,-8,-8,-7,-7,-6,-6,-5,
68                  -5,-5,-4,-4,-3,-3,-3,-2,-2,-1,-1,-0,
69                  0,1,1,2,2,3,3,4,4,5,6,6,
70                  7,7,8,8,9,9,10,11,11,12,12,13,
71                  13,14,14,15,16,16,17,17,18,18,19,19,
72                  20,20,21,21,22,22,23,23,24,24,25,25,
73                  26,26,27,27,28,28,29,29,30,30,31,31,
74                  32,32,33,33,34,35,35,36,36,37,38,38,
75                  39,39,40,41,41,42,42,43,43,44,44,44,
76                  45,45,45,46,46,47,47,47,48,48,48,49,
77                  49,49,50,50,51,51,52,52,53,53,54,54,
78                  55,55,56,56,57,57,58,58,59,59,60,60,
79                  61,61,62,62,63,63,64,64,65,65,66,66,
80                  67,67,68,68,69,69,70,70,71,72,72,73,
81                  73,74,74,75,75,76,77,77,78,78,79,79,
82                  80,80,81,81,82,82,83,83,84,84,84,85,
83                  85,86,86,86,87,87,88,88,88,89,89,90,
84                  90,91,91,92,92,93,93,94,94,95,95,96,
85                  96,97,97,98,98,99,99,100,100,101,101,102,
86                  102,103,103,104,104,105,105,106,106,107,107,108,
87                  108,109,109,110,110,111,111,112,112,113,113,113,
88                  114,114,115,115,116,116,116,117,117,118,118,118,
89                  119,119,120,120,120,121,121,122,122,123,123,124,
90                  124,125,125,125,126,126,127,127,128,128,129,129,
91                  129,130,130,131,131,132,132,133,133,134,134,134,
92                  135,135,136,136,137,137,138,138,138,139,139,139,
93                  140,140,140,141,141,142,142,142,143,143,143,144,
94                  144,144,145,145,146,146,147,147,148,148,149,150,
95                  150,151,151,152,152,153,154,154,155,155,156,156,
96                  157,158,158,159,159,160,160,161,161,162,162,163,
97                  163,164,164,165,165,166,166,167,167,168,168,169,
98                  169,170,170,171,171,172,172,173,173,174,174,175,
99                  175,176,176,177,177,178,178,178,179,179,180,180,
100                 180,181,181,182,182,182,183,183,184,184,184,185,
101                 185,186,186,187,187,188,188,189,189,190,191,191,
102                 192,192,193,193,194,195,195,196,196,197,197,198,
103                 199,199,200,200,201,201,202,202,203,203,204,204,
104                 205,205,206,206,207,207,208,208,209,209,210,210,
105                 210,211,211,212,212,213,213,213,214,214,215,215,
106                 215,216,216,217,217,217,218,218,219,219,219,220,
107                 220,221,221,221,222,222,223,223,223,224,224,225,
108                 225,226,226,227,227,228,228,229,229,230,231,231,
109                 232,232,233,233,234,234,235,235,236,236,237,237,
110                 238,238,239,239,239,240,240,241,241,242,242,243,
111                 243,243,244,244,245,245,246,246,247,247,248,248,
112                 249,249,250,250,251,252,252,253,253,254,254,255,
113                 255,256,257,257,258,258,259,260,260,261,261,262,
114                 263,263,264,264,265,265,266,267,267,268,268,269,
115                 270,270,271,271,272,272,273,273,274,274,275,275,
116                 276,276,277,277,278,278,279,279,280,280,281,281,
117                 282,282,283,283,284,284,284,285,285,286,286,287,
118                 287,288,288,289,289,290,290,291,291,292,292,293,
119                 293,294,295,295,296,296,297,297,298,298,299,299,
120                 300,301,301,302,302,303,304,304,305,305,306,307,
121                 307,308,308,309,310,310,311,311,312,312,313,313,
122                 314,314,315,315,316,316,317,317,317,318,318,319,
123                 319,320,320,321,321,321,322,322,323,323,324,324,
124                 325,325,326,326,327,327,328,328,329,329,330,330,
125                 331,331,332,332,333,333,334,334,335,335,336,336,
126                 337,337,338,338,339,339,340,340,341,341,342,342,
127                 343,343,344,344,345,345,346,346,347,347,348,348,
128                 349,349,350,350,351,351,352,352,352,353,353,354,
129                 354,354,355,355,356,356,356,357,357,357,358,358,
130                 359,359,359,360,360,361,361,362,363,363,364,365,
131                 366,366,367,368,369,369,370,371,372,372,373,374,
132                 374,375,375,376,376,377,377,378,378,379,379,380,
133                 380,381,381,382,382,383,383,384,384,384,384,385,
```

```
                  385,385,385,386,386,386,386,387,387,387,387,387,
                  388,388,388,388,389,389,389,389,390,390,390,390,
                  390,391,391,391,391,392,392,392,392,393,393,393,
                  393,393,394,394,394,394,395,395,395,395,396,396,
                  396,396,396,397,397,397,397,398,398,398,398,399,
                  399,399,399,399,400,400,400,400,401,401,401,401,
                  402,402,402,402,402,403,403,403,403,404,404,404,
                  404,405,405,405,405,405,406,406,406,406,407,407,
                  407,407,408,408,408,408,409,409,409,409,409,410,
                  410,410,410,411,411,411,411,412,412,412,412,412,
                  413,413,413,413,414,414,414,414,415,415,415,415,
                  415,416,416,416,416,417,417,417,417,418,418,418,
                  418,418,419,419,419,419,420,420,420,420,421,421,
                  421,421,421,422,422,422,422,423,423,423,423,424,
                  424,424,424,424,425,425,425,425,426,426,426,426,
                  427,427,427,427,427,428,428,428,428,429,429,429,
                  429,430,430,430 };

String EOL=String(LF)+String(CR);

// SPI library interface
char spi_transfer(volatile char data)
{
  SPDR = data;                    // Start the transmission
  while (!(SPSR & (1<<SPIF)))     // Wait the end of the transmission
  {
  };
  return SPDR;                    // return the received byte
}

// LED display support routines
void write_led_decimals(int value)
{
   digitalWrite(SLAVESELECT, LOW);
   delay(10);
   spi_transfer(0x77);     // Decimal Point OpCode
   spi_transfer(value);    // Decimal Point Values
   digitalWrite(SLAVESELECT, HIGH); //release chip, signal end transfer
}
void write_led_command(int digit1, int digit2)
{
   digitalWrite(SLAVESELECT, LOW);
   delay(10);
   spi_transfer(digit1);    //
   spi_transfer(digit2);    //
   digitalWrite(SLAVESELECT, HIGH); //release chip, signal end transfer
}

void write_led_numbers(int digit1, int digit2, int digit3, int digit4)
{
   digitalWrite(SLAVESELECT, LOW);
   delay(10);
   spi_transfer(digit1);    // Thousands Digit
   spi_transfer(digit2);    // Hundreds Digit
   spi_transfer(digit3);    // Tens Digit
   spi_transfer(digit4);    // Ones Digit
   digitalWrite(SLAVESELECT, HIGH); //release chip, signal end transfer
}
void write_led(unsigned short num, unsigned short base, unsigned short pad)
{
    unsigned short digit[4] = { 0, ' ', ' ', ' ' };
    unsigned short place = 0;

    if ( (base<2) || (base>16) || (num>(base*base*base*base-1)) ) {
        write_led_numbers(' ', 0x00, 0x0f, ' ');  // indicate overflow
    } else {
        while ( (num || pad) && (place<4) ) {
```

```
            if ( (num>0)  || pad )
                digit[place++] = num % base;
            num /= base;
        }
        write_led_numbers(digit[3], digit[2], digit[1], digit[0]);
    }
}


/* itoa:  convert n to characters in s */
 void itoa(int n, char s[])
 {
     int i, sign;

     if ((sign = n) < 0)  /* record sign */
         n = -n;          /* make n positive */
     i = 0;
     do {       /* generate digits in reverse order */
         s[i++] = n % 10 + '0';   /* get next digit */
     } while ((n /= 10) > 0);     /* delete it */
     if (sign < 0)
         s[i++] = '-';
     s[i] = '\0';
     reverse(s);
 }
 /* reverse:  reverse string s in place */
 void reverse(char s[])
 {
     int i, j;
     char c;

     for (i = 0, j = strlen(s)-1; i<j; i++, j--) {
         c = s[i];
         s[i] = s[j];
         s[j] = c;
     }
 }

void slow() {
  //Serial.println("Slow ");
  //digitalWrite(Slow_pin,HIGH);   // Debug, allows setting delay values
    while (Serial.available() > 0) {
    get_cmd();
  }
     switch(index-1) {
       case 3 :  // 3 digit command (100-359)
         write_led_numbers(0x63, cmd[1], cmd[2], cmd[3]);
         break;
       case 2 :  // 2 digit command (10-99)
         write_led_numbers(0x20, 0x63, cmd[1], cmd[2]);
         cAng=cAng/10;
         break;
       case 1 : // single char command (0-9)
         write_led_numbers(0x20, 0x20, 0x63, cmd[1]);
         cAng=cAng/100;
         break;
       default :  //for bumping
         break;
     }
    delay(400);
    index=0;
    ambientLight=analogRead(lightSensor);
//  Serial.println(ambientLight);
    if (ambientLight > 650) {
//    Serial.println("dim");
    write_led_command(0x7a,dim);  // Set brightness= dim
  }
```

```
268     else
269      if (ambientLight < 600) {
270   //     Serial.println("bright");
271        write_led_command(0x7a,bright);  // Set brightness= bright
272     }
273     null=narrow;
274
275
276   //  delay(10);
277     //digitalWrite(Slow_pin,LOW);    // Debug, allows setting delay values
278   }
279
280
281   void get_cmd() {
282     char incomingByte = 0;        // for incoming serial data
283     int incomingbits = 0;
284     int i;
285     byte buffer(5);
286
287   //      String report=String(aAng, DEC)+String(".0")+String(EOL);
288         String report=String("A=")+String(aAng, DEC)+String(".0 S=y M")+String(EOL);
289         incomingByte = Serial.read();
290   //      Serial.print(incomingByte);
291          switch (incomingByte)
292            {
293            case 0x2B :  //"+"
294              cAng=cAng+bump;
295              index=0;
296              break;
297            case 0x2D :  //"-"
298              cAng=cAng-bump;
299              index=0;
300              break;
301            case 0x41 :  //"A"
302              index = 1; cAng = 0;
303              break;
304            case 0x0d :  //Carraige Return
305   //             Serial.print(aAng);    //report current position to PC
306   //             Serial.print(".0");
307   //             Serial.write(10);
308   //             Serial.write(13);
309                  Serial.print(report);
310   //            Serial.print("index = ");
311   //            Serial.println(index);
312   //            Serial.println(" ");
313   //            Serial.print("cAng");
314   //            Serial.print(". ->");
315   //            Serial.println(cAng);
316              Serial.flush();  // should make Serial.available return -1
317              delay(10);
318              break;
319            case 0x30 :
320            case 0x31 :
321            case 0x32 :
322            case 0x33 :
323            case 0x34 :
324            case 0x35 :
325            case 0x36 :
326            case 0x37 :
327            case 0x38 :
328            case 0x39 :
329              cmd[index] = incomingByte - 0x30;
330              incomingbits = incomingByte - 0x30;
331              cAng = cAng + incomingbits*sf[index];
332              index++;
333              break;
334           default :
```

```
335             Serial.flush();  // incomingByte not a number, cancel job
336   //         }
337         }
338         null=narrow;
339   }
340
341   void setup() {
342     Serial.begin(9600);
343   //  Serial.println("Initializing");
344     // SPI &  LED display setup
345     byte clr=0;
346     pinMode(DATAOUT, OUTPUT);
347     pinMode(DATAIN, INPUT);
348     pinMode(SPICLOCK, OUTPUT);
349     pinMode(SLAVESELECT, OUTPUT);
350   // pinMode(Fast_pin, OUTPUT);
351   // pinMode(Slow_pin, OUTPUT);
352     digitalWrite(SLAVESELECT, HIGH); //disable device
353   //  SPCR = 01010010;
354     //interrupt disabled,spi enabled,msb 1st,master,clk low when idle,
355     //sample on leading edge of clk,system clock/64
356     SPCR = (1<<SPE)|(1<<MSTR)|(1<<SPR1);
357     clr=SPSR;
358     clr=SPDR;
359   //  digitalWrite(Fast_pin,LOW);    // Debug, allows setting delay values
360   //  digitalWrite(Slow_pin,LOW);    // Debug, allows setting delay values
361   //  delay(100);
362     write_led_numbers(0x78,0x78,0x78,0x78); //Blank display
363   //  delay(50);
364     write_led_decimals(0x00); // All decimal points off
365     write_led_command(0x7a,0x0f);  // Set brightness= bright
366     null=narrow;
367   //  Serial.println("Done");
368   }
369
370
371   void loop() {
372
373     if(state==0) {
374       slow();
375     }
376      else  {
377      //digitalWrite(Fast_pin,HIGH);  // Debug, allows setting delay values
378      // read the Potentiometer and display the actual rotor angle:
379        sensorValue = analogRead(analogInPin);
380      //Serial.println(sensorValue);
381       // map it to the range of the analog out:
382        aAng = pgm_read_word_near(a2d10+sensorValue);
383        //Serial.println(aAng);
384         write_led(aAng,10,0);
385
386         CW_state = digitalRead(CW_pin);
387         CCW_state = digitalRead(CCW_pin);
388         if ((CW_state == HIGH) && (CCW_state == HIGH)) {
389         // calculate the motor command and close the loop
390           ang_error=aAng-cAng;
391           calcOutput = ang_error*Kpos;
392           if (abs(ang_error) <= null) { // done
393             calcOutput = 0;
394             null=wide;
395           }
396           if (ang_error > 0) {  // not done
397             polarity = HIGH;
398           }
399           else {
400             polarity = LOW;
401             calcOutput = -calcOutput;
```

```
          }
          if (calcOutput > 255) {
            calcOutput = 255;
          }
          digitalWrite(PolarityPin, polarity);
          cmdOutput=int(calcOutput);  // PWMA requires integer not double
          analogWrite(PWMA_pin, cmdOutput);
          delay(40);
        }

        if (CW_state == LOW) {
        // Serial.println("CW trim");
            polarity = LOW;
            digitalWrite(PolarityPin, polarity);
            analogWrite(PWMA_pin, 255);
            delay(40);
            cAng=aAng;
        }

        if (CCW_state == LOW) {
        // Serial.println("CCW trim");
            polarity = HIGH;
            digitalWrite(PolarityPin, polarity);
            analogWrite(PWMA_pin, 255);
            delay(40);
            cAng=aAng;
        }

     if(state==15) {
        state=-1;
       }
    }
    state++;
//   Serial.print(state);

}

```