```
/*********************************************************************
 * Project name:  Polarity Rotator - Two 7-Segment Display Multiplexing  12-APR-2021
 * polrotW.c
 * Uses interrupt on RA2, trailing edge trigger on CLK signal from LS7184
 * Looks for direction signal from LS7184 on RA1, clockwise = HIGH
 * Rotor quadrature encoder outputs 360 pulses for 360 degree rotation
 * Program displays pulses to +/- 99 degrees from horizontal (0 degrees)
 * Corrected for int(count) and use 'count' to shut of motors in Park mode
 * Added new TX routines
 *********************************************************************/
#define _LEGACY_HEADERS                 //Added for compiler versions 9.81+
#include <pic.h>                        // Include HITECH CC header file
#include <stdio.h>                      //
#include <stdlib.h>                     // provides ABS command
//Internal clock, Watchdog off, MCLR off, Code Unprotected
__CONFIG (INTIO & WDTDIS & PWRTEN & MCLRDIS & BORDIS & UNPROTECT & IESODIS & FCMDIS );
unsigned short i, DD0, DD1;
int x;
int c;
unsigned int delay;

int count;          // count is actual number of pulses (position in degrees from horizontal)
int t;              // t is calculated transmit angle
int d;              // d = Geometric angle
int r;              // r = Best received angle
int epsilon = 4;    // compensate for motor overshoot
int park;           // park = 0 means don't park
int RX;             // RX = 1 means you are receiving

unsigned short mask(unsigned short num);
unsigned short mask1(unsigned short num1);
void pause( unsigned short msvalue );   //Establish pause function
void msecbase( void );                  //Establish millisecond base function
void goCW(void);                        //Motor start CW
void goCCW(void);                       //Motor start CCW
void main()
{
  ANSEL = 0;              // Intialize A/D ports off
  ANSELH = 0;             // Intialize A/D ports off
  CM1CON0 = 0;            // Initialize Comparator 1 off
  CM2CON0 = 0;            // Initialize Coparator 2 off
  TRISA = 0xFF;           // All PortA I/O inputs
  PORTA = 0;              // Clear PortA
  TRISA2 = 1;             // Set A2/INT pin to input
  TRISB = 0x00;           // Set PORTB direction to be output
  PORTB = 0xff;           // Turn OFF LEDs on PORTB
  TRISC = 0x00;           // All PortC I/O outputs
  PORTC = 0xff;           // Set PortC port to turn OFF 7 segments
  RABPU = 0;              // Enable internal pullups on selected pins
  WPUA5 = 1;              // Enable pull-up on Z-pulse input
  WPUA4 = 1;              // Pull up for Switch RA4, RA3 is external PU
  WPUA2 = 1;              // Enable pull-up on interrupt input
  WPUA0 = 0;              // Ensure no pull-up on RA0
  INTEDG = 1;             // leading edge (low going) triggers the interrupt
  INTE = 1;               // enable the external interrupt
  GIE = 1;                // Global interrupt enable
  RB6 = 1;                // CCW, counterclockwise relay, 1 = de-energized
  RB7 = 1;                // CW, clockwise relay, 1 = de-energized

do
 {
x = count;                // convert 360 count to degrees
c = abs(x);               // eliminate sign just for display purposes
//*****************************************************
//TX routine
//*****************************************************
 {
if (RA4==0)               // test GEO switch
    d = count;            // store geometric polarity

if (RA3==0)               // test RX switch
   r = count;             // store receive polarity

 t = 2*d - r;             // compute TX polarity
```

```
 if (t > 99) t = t - 180;   // These lines keep T in top 'quadrants'
 if (t < -99) t = t + 180;

 if (RA0==1)                // test for transmit
 {
  if (abs(t - count) > epsilon) //transmit, if not close, rotate ccw or cw
      { if(t < count)           // if t < count, go cw
        goCW();
            else
        goCCW();
      }
  }
//************ RX Routine **************
  else                          // not transmitting, return to receive
  if (RX == 1 && RA0 == 0)
  {
   if (abs(r-count) > epsilon)   // receive, if not close, rotate
       {if(r < count)            // if r < count, go CW
           goCW();
           else
           goCCW();
       }
  }
  }                             // end T/R signal processing

//************ Park Routine **************
 {
        if (RA3 == 0 && RA4 == 0)   // Press Geo and RX at same time to park
                park = 1;

        if (park == 1)
    {   if (count < 0)          // If count is less than 0 activate CW relay
            goCCW();
         if (count > 0)         // If count is greater than 0 activate CCW relay
            goCW();
    }
 }
//*****************************************************
//  Display Loop
//*****************************************************

 if (count<0)                   // Test for CCW (count less than zero)
  {
  DD0 = c%10;                   // Extract Ones Digit
  DD0 = mask1(DD0);
  DD1 = (c/10)%10;              // Extract Tens Digit
  DD1 = mask1(DD1);
  }
 else                           // lights the minus sign
  {
  DD0 = c%10;                   // Extract Ones Digit
  DD0 = mask(DD0);
  DD1 = (c/10)%10;              // Extract Tens Digit
  DD1 = mask(DD1);
  }
 for (i = 0; i<=10; i++)
  {
      PORTC = DD0;
      RB4 = 0;                  // Select Ones Digit
      RB5 = 1;
      pause(5);
      PORTC = DD1;
      RB4 = 1;
      RB5 = 0;                  // Select Tens Digit
      pause(5);
  }                             // end for

 } while(1);                    // endless loop
}                               // End main
//*****************************************************
//Interrupt routine
//*****************************************************
void interrupt isr(void)            // This is the interrupt service routine
```

```
{
        if(INTF)                    // Test for interrupt
     {
        if (RA2==0 && RA1 == 0)     // Test for  coincidence of clock and direction
              count = count - 1;    // CCW
          else
              count = count + 1;    // CW

        if (RA5 == 0)               // Check for Z-pulse
              count = 0;            // Reset count to correct for errors
//***********  Stop motor at park position (zero)
              if (park == 1)
                 if (count == 0)       //  Stop motor on zero
              {       RB6 = 1;         //  High turns off relays
                      RB7 = 1;
                      park = 0;        // Clear the park flag
              }
//************  TX Stop Routine
        {if (RA0==1 && t==count)        // test for transmit
    {
           {RB6=1;
            RB7=1;                     // STOP
            RX = 1;
           }
    }
//************ RX stop routine ******************
        if (RX==1 && r==count)          //
           { RB6=1;
             RB7=1;                    // STOP
             RX = 0;                   // CLEAR TX FLAG, DISABLES
           }                           // AUTO UNTIL THE NEXT RA0=1
        }
        INTF=0;                         // clear the interrupt
        }
}
//*****************************************************
//pause - multiple millisecond delay routine
//*****************************************************
void pause( unsigned short msvalue )
{
        unsigned short p;
        for (p=0; p<=msvalue; p++)      //Loop through a delay equal to usvalue
              {                         // in milliseconds.
              msecbase();               //Jump to millisec delay routine
              }
}
//*****************************************************
//msecbase - 1 millisecond pause routine
//*****************************************************
void msecbase(void)
{
        OPTION = 0b00000001;            //Set prescaler to TMRO 1:4
        TMR0 = 0xd;                     //Preset TMRO to overflow on 250 counts
        while(!T0IF);                   //Stay until TMRO overflow flag equals 1
        T0IF = 0;                       //Clear the TMR0 overflow flag
}
//*****************************************************
//mask - Function to Return mask for common anode 7-seg. display
//*****************************************************
 unsigned short mask(unsigned short num)
{
 switch (num)
 {
 case 0 : return 0x82;
 case 1 : return 0xFA;
 case 2 : return 0xA1;
 case 3 : return 0xB0;
 case 4 : return 0xD8;
 case 5 : return 0x94;
 case 6 : return 0x84;
 case 7 : return 0xBA;
 case 8 : return 0x80;
 case 9 : return 0x98;
 }                                      //case end
```

```
}
 unsigned short mask1(unsigned short num1) //These cases light the minus sign
{                                          // for 0 to -99 deg
 switch (num1)
 {
 case 0 : return 0x02;
 case 1 : return 0x7A;
 case 2 : return 0x21;
 case 3 : return 0x30;
 case 4 : return 0x58;
 case 5 : return 0x14;
 case 6 : return 0x04;
 case 7 : return 0x3A;
 case 8 : return 0x00;
 case 9 : return 0x18;
 }                                         //case end
}
//*****************************************************
//goCW() - rotates clockwise
//*****************************************************
void goCW(void)
{
  RB6 = 1;                // CCW counterclockwise relay, 1 = de-energized
  RB7 = 0;                // CW, clockwise relay, 0 = energized
}
//*****************************************************
//goCCW() - rotates counterclockwise
//*****************************************************
void goCCW(void)
{
  RB6 = 0;                // CCW counterclockwise relay, 0 = energized
  RB7 = 1;                // CW, clockwise relay, 1 = energized
}
```