TERRAN BLAKE // ECE 241 // WEDNESDAY 7:30 LAB

```
#include <math.h>
#include<LiquidCrystal.h>

LiquidCrystal LcdDriver(11, 9, 5, 6, 7, 8);

// State machine and data for decoding a stream of serial data into a float.
enum FloatingPointDecode { IntegerDecode, FractionalDecode, ExponentDecode };
FloatingPointDecode DecodeState = IntegerDecode;
float IntermediateFloat;
bool NegativeFlag;
float FractionalPosition;
bool NegativeExponent;
int Exponent;
bool InputReceived;
int counter = 0;
char operation = ' ';

void LcdPrintFloat( float Input, int digits );

// When DecodeFloat returns a true the value entered appears
// in ResultingFloat
float ResultingFloat = 0;

// This function resets the decoding process
// so it is clear to receive another number.
void DecodeReset() {
  IntermediateFloat = 0.0;
  FractionalPosition = 0.1;
  NegativeFlag = false;
  NegativeExponent = false;
  Exponent = 0;
  InputReceived = false;
  DecodeState = IntegerDecode;

} // End of DecodeReset()

// Function to process characters as floating point number come in.
// Supports the format +/-III.FFFe+/-EXPONENT
int DecodeFloat( char Ch ) {
  // check to see that incoming character can be
  // part of a floating point number,
  if (  isDigit( Ch )  // such as '0' to '9',
        || Ch == '.'   // decimal point,
        || Ch == '+'   // plus sign,
        || Ch == '-'   // minus sign,
        || (Ch | 0x20) == 'e' ) // or exponent indicator.
```

```
 { // upper or lower case E

   InputReceived = true;
   // based on state, apply the incoming character.
   switch ( DecodeState ) {
    case IntegerDecode:    // reading in first part of significand
     if ( isDigit( Ch ) )  // digit coming in.
      IntermediateFloat = 10 * IntermediateFloat + ( Ch - '0' );
     else if ( Ch == '-'              // Negative sign
            && IntermediateFloat == 0 ) // at start string
      NegativeFlag = true;             // indicates a negative.
     else if ( Ch == '.' ) // Decimal point
      DecodeState = FractionalDecode; // move to reading fractional part.
     else if ( (Ch | 0x20) == 'e' ) // Change to lower case and if e
      DecodeState = ExponentDecode;   // move to reading exponent.
     break;
    case FractionalDecode: // Here system reading fractional part.
     if ( isDigit( Ch ) ) {
      IntermediateFloat += FractionalPosition * ( Ch - '0' );
      FractionalPosition *= 0.1; // move down a digit.
     }
     else if ( (Ch | 0x20) == 'e' )// Change to lower case and if e
      DecodeState = ExponentDecode; // move to reading exponent.
     break;
    case ExponentDecode: // Reading in Exponent.
     if ( isDigit( Ch ) )
      Exponent = 10 * Exponent + ( Ch - '0' ); // add in digit.
     else if ( Ch == '-'  // if negative sign
            && Exponent == 0 ) // at start of exponent
      NegativeExponent = true;   // set for negative sign.

   }
  }
  else if ( InputReceived ) { // If we actually have received some characters.
   // Generate the resulting number
   if ( NegativeFlag )
    ResultingFloat = -IntermediateFloat
                * pow( 10, NegativeExponent ? -Exponent : Exponent );
   else
    ResultingFloat = IntermediateFloat
                * pow( 10, NegativeExponent ? -Exponent : Exponent );
   return Ch; // Return terminator to calling function.
  }
  return 0; // Return a false as the default.

}// End of DecodeFloat

void LcdPrintFloat( float ResultingInput, int digits ) {
```

```
  int Expon = 0;
  int sign = 1;

  if ( ResultingInput < 0 ) {
    ResultingInput = -ResultingInput;
    sign = -1;

  }
  while ( ResultingInput >= 10 ) {
    Expon++;
    ResultingInput /= 10;

  }
  while ( ResultingInput < 1 ) {
    Expon--;
    ResultingInput *= 10;

  }
  LcdDriver.print ( sign * ResultingInput );

  if ( Expon != 0 ) {
    LcdDriver.print( "e" );
    LcdDriver.print( Expon );

  }
} // End of LcdPrintFloat

// put your setup code here, to run once:
void setup() {
  Serial.begin(38400); // Set up Serial port.
  DecodeReset();      // Start in a initialized state.

} // End of setup.

// put your main code here, to run repeatedly:
void loop() {
  // Check for incoming character.
  if ( Serial.available() ) {
    // Read in and process character
    if ( DecodeFloat( Serial.read() ) ) {
      // if a true is returned,
      // the end of the number was reached
      // and we can print it out.
      //Serial.println( ResultingFloat, 5 );
      LcdPrintFloat(ResultingFloat, 5 );
      DecodeReset(); // Reset for next number.
      LcdDriver.setCursor(1,0);
```

```
  } // End of test for number complete.

  //char operation = Serial.read();

  switch( CalculatorState ) {
   case FirstNumber:
    if( DecodeFloat( Ch ) ) {
     // DecodeFloat returns true if end of number detected.
     // So save and display number
     // Move to SecondNumber
    }
    break;
   case SecondNumber:
     /*

        This is as far as I got before leaving the Office Hours on Monday.
        When I got home I couldn't get anything to compile and my Arduino
        wasn't being recognized by my computer.

          */
  }

 } // End of loop

}
```