```
/* This program displays the Character Speed WPM on line 1 and
Character Clock Timing on line 2 of  a 16 x 2 LCD display based on a
HD44780 display controller (Parallel Interface). LCD fed from the MEGA
End pins.
/
ddf - WA7RSO - 07/01/2013 - Modified 08/22/2013
/
Clock Rate in Hz is determined by WPM/1.2  or Timing Interval (in
milliseconds) by 1.2/WPM
Checks for Dit Paddle or Dah Paddle inputs, to execute Dit/Dah Timing.

Modes are:
PARIS Calibration Test
Basic Self-Completing Dits & Dahs,
Iambic,
Bug-Simulation,
Squeeze-Key,
Simple Key Memory

Send_DIT() and Send_DAH() have been shortened to DIT() and DAH()
*/

#include <LiquidCrystal.h>  //  alert compiler to include the lcd library

// initialize the library with the numbers of the Arduino pins used
byte LCDGround1=54;       // Pin 1 on the LCD
byte LCD5V =52;           // Pin 2 on the LCD
byte Contrast =50;        // Pin 3 on the LCD
byte ReadStrobe=48;       // Pin 4 on the LCD
byte ReadWrite=46;        // Pin 5 on the LCD
byte Enable=44;           // Pin 6 on the LCD
byte DB0=42;            // Pin 7 on the LCD
byte DB1=40;            // Pin 8 on the LCD
byte DB2=38;            // Pin 9 on the LCD
byte DB3=36;            // Pin 10 on the LCD
byte DB4=34;              // Pin 11 on the LCD
byte DB5=32;              // Pin 12 on the LCD
byte DB6=30;              // Pin 13 on the LCD
byte DB7=28;              // Pin 14 on the LCD
byte Backlight=26;        // Pin 15 on the LCD
byte LCDGround2=24;       // Pin 16 on the LCD
// Using "Equated" values (lcd(48,44,34,32,30,28))
LiquidCrystal lcd(ReadStrobe,Enable,DB4,DB5,DB6,DB7);
//i.e. LiquidCrystal lcd(48,44,34,32,30,28);
//
int WPM = 15;                    // All calculations and 2-line display
are from this setting
float Speed = WPM/1.2;            // Speed in Clocks per Second (Hz)
float Clock = (1000*(1.2/WPM));    // Interval in milliseconds
// Physical Pin Assignments
byte TP=8;            // Dit Paddle Pin
byte HP=9;            // Dah Paddle Pin
//pinMode (TP, INPUT_PULLUP);  // Turn on the 20K pullup resistor
```

```
//pinMode (HP, INPUT_PULLUP);
// Internal Logic for Modes and Controls
int TR=0;            // Wait for a "Request"
int HR=0;            // Wait for a "Request"
byte TE=1;            // Initialize as "Enabled"
byte HE=1;            // Initialize as "Enabled"
// Resulting Outputs
int TA=11;           // Dit Active Pin Assignment
int HA=12;           // Dah Active Pin Assignment
int Xmit=13;         // Xmit Active Pin Assignment, by either TA or HA
Action
// "Mode" Definitions:
// Mode-0  PARIS Test Demo Mode
// Mode-1  Standard Self-Completing
// Mode-2  IAMBIC
// Mode-3  BUG
// Mode-4  Squeeze
// Mode-5  Straight Key
//
String Mode_Text;
//= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= =
int Mode=0;          // Self-Completing, with "Squeeze Hold"
//= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =
= =
int dits,dahs,count;

int Normal_Flag, Iambic_Flag, Bug_Flag, Squeeze_Flag;

String CharSpeed = ("Char Speed ");      // i.e. "Words per Minute"
String CharClock = ("   ms     Hz Clk");
//=====================================================================
void setup()                          //required function
{
  Serial.begin(9600);            // For the "Monitor Feedback"
// === Define the pin assignments ===
 pinMode(LCD5V,OUTPUT);            // Set the +5V Source High
 pinMode(Contrast,OUTPUT);      // Set High
 pinMode(ReadWrite,OUTPUT);      // Set Low
 pinMode(Backlight,OUTPUT);     // Set High
 pinMode(LCDGround2,OUTPUT);    // Set Low
 //
  pinMode(Xmit,OUTPUT);          // Clock Rate for "Dit-On" & "Dit-Off"
  pinMode(TA,OUTPUT);
  pinMode(HA,OUTPUT);
  pinMode(8,INPUT);              // Pin #8 for Input "TP"
  pinMode(9,INPUT);              // Pin #9 for Input "HP"
  pinMode(10,OUTPUT);

// Initial LCD Display of Character WPM and correct timing required
  lcd.begin(16,2);              //let the program know the size of the
display to be handled
//
  digitalWrite(LCD5V,HIGH);            // Turn on the +5V Sourc
```

```
  digitalWrite(Contrast,LOW);        // Set the Contrast for normal
//(Manual ground, or a diode, appears to be necessary on the Blue 2x16
LCDs, for LOW on R/W)
  digitalWrite(ReadWrite,LOW);        // Eanable Writing to the LCD
  digitalWrite(Backlight,HIGH);      // Turn ON the Backlight
  digitalWrite(LCDGround2,LOW);      // Establish a Low to simulate
Ground on pin 16
//
  printspeed();                  // calls the function to print the WPM
message
  printspecs();                  // calls the function to print the
"Timing Parametes"
  delay(3000);

//   Show_Mode();

/*
  if(Mode==5) {
    Straight_Key(); }
*/

//  printclock();                  // calls the function to print the
first 16 characters of the message
}
//======================== Now, Run the Respective "Mode" required
================================================
void loop()
{
  if(Mode==0) {
    PARIS_Calibration(); }
  if(Mode==1) {
    Self_Completing(); }
  if(Mode==2) {
    Iambic(); }
  if(Mode==3) {
    Bug_Emulation(); }
  if(Mode==4) {
    Squeeze(); }
}
//======================== Mode-0 "PARIS" Calibration
================================================
void PARIS_Calibration()               //  Dummy Loop, or "PARIS"
Calibration
{
 loop;
 {
  count=1;
  for(int i=0; i<WPM; i++)
   {
    PARIS();                    // Invoke the "PARIS" Calibration Routine
for this WPM
   }
   delay(500);
     printspecs();
```

```
  delay(3000);
 }
}
//====================== LCD 1st Line - "WPM"
================================================
void printspeed()                   // Show the Character Speed in WPM
  {
    lcd.setCursor (0,0);             // Set cursor at beginning of 1st
Row ("0")
    lcd.print(CharSpeed);            // print the "Character Speed "
    lcd.print(WPM);
    lcd.print("WPM");
//    Serial.println(CharSpeed);
  }
//====================== LCD 2nd Line - Timing
================================================
void printspecs()
  {
    lcd.setCursor(0,1);               //sets cursor at the bginning of
the second line and then moves right
    lcd.print(Clock);
    lcd.print("ms  ");
    lcd.print(Speed);
    lcd.print("Hz");
//    lcd.print(CharClock);
//    Serial.println(CharClock);
  }

//======================== Send a Dit by turning #13 LOW,  and Indicate
by #11 Low ======================================
void DIT(int dits){
 if(Mode != 0)
  {
  Clear2ndRow();
   lcd.setCursor(0,1);
   lcd.print("Sending Dit");
  }
 for(int t=0; t<dits; t++)          // Send this many "Timed Dits"
  {
  digitalWrite(Xmit,HIGH);          // Turn on the Xmtr
  digitalWrite(TA,HIGH);            // Dit Indicator
  delay(Clock);
  digitalWrite(Xmit,LOW);            // Turn off the Xmtr
  digitalWrite(TA,LOW);              // Dit Indicator
  delay(Clock);
  }
}
//======================== Send a Dah by turning #13 LOW,  and Indicate
by #12 Low ======================================
void DAH(int dahs){
 if(Mode !=0)
  {
  Clear2ndRow();
   lcd.setCursor(0,1);
```

```
   lcd.print("Sending Dah");
  }
 for(int t=0; t<dahs; t++)          // Send this many "Timed Dahs"
  {
  digitalWrite(Xmit,HIGH);          // Turn on the Xmtr
  digitalWrite(HA,HIGH);            // Dah Indicator
  delay(3*Clock);
  digitalWrite(Xmit,LOW);           // Turn off the Xmtr
  digitalWrite(HA,LOW);             // Dah Indicator
  delay(Clock);
  }
}
//================ Composition of "PARIS" Code Characters
=====================================================
void PARIS()                 // "PARIS", for calibration
{
  DisplayParis();                      // Display "PARIS" on the LCD

  DIT(1); DAH(2); DIT(1);      // "P"
    delay(2*Clock);                                  // 2 Clocks Spacing
  DIT(1); DAH(1);                   // "A"
    delay(2*Clock);                                  // 2 Clocks Spacing
  DIT(1); DAH(1); DIT(1);      // "R"
    delay(2*Clock);                                  // 2 Clocks Spacing
  DIT(2);                               // "I"
    delay(2*Clock);                                  // 2 Clocks Spacing
  DIT(3);                               // "S"
    delay(5*Clock);
}
//===================== LCD "PARIS" Calibration Display
================================================
void DisplayParis()
{
 Clear2ndRow();
// delay(250);                        // Delay 1/4 second between display
of "PARIS"
 lcd.setCursor(0,1);
 lcd.print(count);
 count ++;
 lcd.print(" PARIS");
}
//===================== "Self-Completing" works good - 06/28/2013
============================================
void Self_Completing()              // Stays here, unless involved with
Squeeze
{
 Normal_Flag=1;
  Mode_Text = ("Self Complete");
  Show_Mode();
//  Clear2ndRow();
//  Check_Paddles();
 if(digitalRead(TP)==HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
```

```
 if(digitalRead(TP)==HIGH && digitalRead(HP)!=HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
 if(digitalRead(TP)!=HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
loop;
 {
 while(digitalRead(TP)==LOW && digitalRead(HP)!=LOW)
   {
   DIT(1);                      // Invoke sending "Timed Dits"
   }
 while(digitalRead(HP)==LOW && digitalRead(TP)!=LOW)
   {
   DAH(1);                      // Invoke sending "Timed Dahs"
   }
  if(digitalRead(TP)==LOW && digitalRead (HP)==LOW && Squeeze_Flag)
  {
  Squeeze();
  }
 }
/*
  lcd.setCursor(0,1);
  lcd.print("Exiting");
  delay(500);
*/
}
//==================== "IAMBIC" Mode - Mode #2 ========= Checked
06/28/2013 ==================================
void Iambic()                           // Toggle Dits & Dahs if both are
requested.
{
 Iambic_Flag=1;
  Mode_Text = (" IAMBIC Mode");
  Show_Mode();

//  Check_Paddles();
 if(digitalRead(TP)==HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
 if(digitalRead(TP)==HIGH && digitalRead(HP)!=HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
 if(digitalRead(TP)!=HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"

 if(digitalRead(TP)==LOW && digitalRead(HP)!=LOW && TE==1)
 {
   DIT(1);                      // Invoke sending "Timed Dits"
 }
 if(digitalRead(HP)==LOW && digitalRead(TP)!=LOW && HE==1)
 {
  DAH(1);                      // Invoke sending "Timed Dahs"
 }
 if(digitalRead(TP)==LOW && digitalRead(HP)==LOW)   // Check to see if
Dit is enabled (for taking turns)
 {
  if(TE==1){
```

```
      DIT(1);
    TE = !TE;
    HE = 1;
/*
   Clear2ndRow();
   lcd.setCursor(0,1);
   if(HE==1) { lcd.print("DAH is next"); }
   delay(6000);
*/
    }
  if(HE==1) {
      DAH(1);
    HE = !HE;
    TE = 1;
/*
   Clear2ndRow();
   lcd.setCursor(0,1);
   if(TE==1) { lcd.print("DIT is next"); }
   delay(6000);
*/
      }
 }
}
//============== "Bug Emulation" looks good ==== 06/28/2013
================================================
void Bug_Emulation()
{
 Bug_Flag=1;
  Mode_Text = ("Emulate Bug");
  Show_Mode();
  Clear2ndRow();
//  Check_Paddles();
// if(digitalRead(TP)==HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
// if(digitalRead(TP)==HIGH && digitalRead(HP)!=HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
// if(digitalRead(TP)!=HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
  loop;
 {
 while(digitalRead(TP)==LOW && digitalRead (HP)!=LOW)
 {
  DIT(1);                      // Self-Completing Continous Dits
 }
 if(digitalRead(HP)==LOW && digitalRead (TP)!=LOW)
  {
  digitalWrite(Xmit,HIGH);          // Turn on the Xmtr
  digitalWrite(HA,HIGH);            // Dah Indicator
   Clear2ndRow();
   lcd.setCursor(0,1);
   lcd.print("Sending Dah");
  delay(3*Clock);                   // Minimum of 3 time frames
  }
```

```
 while(digitalRead(HP)==LOW && digitalRead (TP)!=LOW)
// Hold the Dah as long as needed
  {
  digitalWrite(Xmit,HIGH);          // Turn on the Xmtr
  digitalWrite(HA,HIGH);            // Dah Indicator
  }
  digitalWrite(Xmit,LOW);           // Turn off the Xmtr
  digitalWrite(HA,LOW);             // Dah Indicator
  delay(Clock);
 }

}
//=====================================================================
void Squeeze()
{
 Squeeze_Flag=1;
  Mode_Text = ("Squeeze Mode");
  Show_Mode();
  Clear2ndRow();
//  Check_Paddles();
 if(digitalRead(TP)==HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
 if(digitalRead(TP)==HIGH && digitalRead(HP)!=HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
 if(digitalRead(TP)!=HIGH && digitalRead(HP)==HIGH) {TE=1; HE=1;}  //
Reset the "Enable Flags"
 loop;
 {
 if(digitalRead(TP)==LOW && digitalRead(HP)==LOW)
   {
   digitalWrite(Xmit,HIGH);          // Turn on the Xmtr
   digitalWrite(TA,HIGH);            // Dit Indicator
   digitalWrite(HA,HIGH);            // Dah Indicator
   delay(3*Clock);                   // Minimum of 3 time frames
   }
  while(digitalRead(TP)==LOW && digitalRead(HP)==LOW)   // Hold as long
as desired
   {
   digitalWrite(Xmit,HIGH);          // Turn on the Xmtr
   digitalWrite(TA,HIGH);            // Dit Indicator
   digitalWrite(HA,HIGH);            // Dah Indicator
   }

   digitalWrite(Xmit,LOW);           // Turn off the Xmtr
   digitalWrite(TA,LOW);             // Dit Indicator
   digitalWrite(HA,LOW);             // Dah Indicator
   delay(Clock);                     // Normal time space
 }
  lcd.setCursor(0,1);
  lcd.print("Now, Act normal");

   Self_Completing();                // Now, act normal
/*
  lcd.setCursor(0,1);
```

```
  lcd.print("Exiting");
  delay(500);
*/
}
//=================== LCD 1st Line - Show the "Mode" Requested
================================================
void Show_Mode()
 {
  Clear1stRow();
  lcd.setCursor(0,0);
  lcd.print(Mode_Text);
//  delay(1000);
 }
//==================== LCD 2nd Line - Show "Action" involved
================================================
void TAG()
{
 Clear2ndRow();
// lcd.setCursor(0,1);
 lcd.print(" ?? ");
 delay(1000);
}
//=============================================================================
void Clear1stRow()
{
 lcd.setCursor(0,0);
 lcd.print("                ");
}
//=============================================================================
void Clear2ndRow()
{
 lcd.setCursor(0,1);
 lcd.print("                ");
}
//=================== Check and spell out if either Dit or Dah Paddles
are High or Low ======================
void Check_Paddles()
{
  Clear2ndRow();
  lcd.setCursor(0,1);
  lcd.print("Checking for Dit");
  delay(1000);

  Clear2ndRow();
  lcd.setCursor(0,1);
  if (digitalRead(TP)==LOW) {
     lcd.print("Dit Paddle LOW");}
  if (digitalRead(TP)==HIGH) {
     lcd.print("Dit Paddle HIGH");}
  delay(1000);

  Clear2ndRow();
  lcd.setCursor(0,1);
  lcd.print("Checking for Dah");
```

```
  delay(1000);

  Clear2ndRow();
  lcd.setCursor(0,1);
  if (digitalRead(HP)==LOW) {
     lcd.print("Dah Paddle LOW");}
  if (digitalRead(HP)==HIGH) {
     lcd.print("Dah Paddle HIGH");}
  delay(1000);
}
//=========================================================================
/*
void loop()
{

}
*/
```