```
/*
* IR Beam Detected Speed Trap (Single Track Version) Version 1.2 LCD Edition
*
* Provided under a Creative Commons Attribution, Non-Commercial Share Alike,3.0 Unported License
*
* COPYRIGHT 2013 - S.D. "Hoffy" Hofmeister, et al
*********************************************************************************************
* CREDITS:
*
* Speed calculations & initial concept coding by: Toni Ryan
*
* IR Beam Detection & LCD Integration by: S.D. "Hoffy" Hofmeister
*
* LCD Library (lcdLib.h & lcdLib.c) The University of Texas at El Paso -
* College of Engineering: http://www.ece.utep.edu/courses/web3376/Lab_5_-_LCD.html
* Elliott Currola, Luis Carlos Bañuelos-Chacon, Elias N Jaquez
*
* This code is a group source project through The Launchpad for Model Railroading Project
* - http://launchpad4mrr.blogspot.com/
*
*********************************************************************************************
* TARGETED TO MSP430 LANUCHPAD W/MSP430G2553 PROCESSOR
*
* Design Notes:
*
* This code is designed to to trigger a speed calculation for a model train passing through 2 IR beam Sensors.
* The code is designed to take a "speed reading" or "Time Count" in both directions.
*
* Distance between Tip of IR Emitter and Tip of Detector has only been tested up to 3.5 Inches under
* incandescent and fluorescent lighting conditions with no failures.
*
* LCD Coding is designed for a QC1602A Ver 2.0 LCD Module
*
* Circuit Pinout:
* PIN 1.0 = RED INDICATOR LED
* PIN 1.1 = UNASSIGNED - UART
* PIN 1.2 = UNASSIGNED - UART
* PIN 1.3 = UNASSIGNED
* PIN 1.4 = Cathode of IR Beam Receiver #1 > Anode to Ground
* PIN 1.5 = Cathode of IR Beam #2 > Anode to Ground
* PIN 1.6 = GREEN INDICATOR LED
* PIN 1.7 = UNASSIGNED
* PIN 2.0 = LCD D4
* PIN 2.1 = LCD D5
* PIN 2.2 = LCD D6
* PIN 2.3 = LCD D7
* PIN 2.4 = E (Enable)
* PIN 2.5 = RS (Register Signal)
*
* PINS 1.1, 1.2, 1.5, 1.7 are left open for integration into other projects
*
* Note Anodes for the IR Emitters connect to VCC and Cathodes to Ground
*/
//*********************************************************************************************
#include <msp430g2553.h>
#include "lcdLib.h"
// Declare functions
void delay ( unsigned int );  // delay for xx 10ms increments
// Declare variables
float scale = 160;    // denominator of scale (N scale 1:160)
float gate = 12;      // timing gate spacing (in inches)
float speed = 0; // speed in MPH
long unsigned int factor = 109090;     // number of ms to travel "gate" inches at 1 MPH for "scale" scale (1 MPH slowest speed we can measure)
long unsigned int counter = 0;     // millisecond counter
int counter_active = 0; // flag to increment counters or stop counting
int output_count = 0;     // flag to display time
int lockgate_1 = 0;   // Locks Gate #1 from interrupting until unlocked
int lockgate_2 = 0; // Locks Gate #2 from interrupting until unlocked
void main(void)
{
  WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  BCSCTL1 = CALBC1_1MHZ; // Set range
  DCOCTL = CALDCO_1MHZ; // Set DCO step + modulation
  CCTL0 = CCIE;                             // CCR0 interrupt enabled
  TACTL = TASSEL_2 + MC_1 + ID_3;           // SMCLK/8, upmode
  CCR0 =  125;                                // 1000 Hz = 1 ms clock

```

```
78      P1DIR |= BIT0;                    // Port 1 P1.6 (Indicator #1) as output
79      P1OUT &= ~BIT0;                     // Port 1 P1.6 (Indicator #1) Set to off State
80      P1DIR |= BIT6;                    // Port 1 P1.7 (Indicator #1) as output
81      P1OUT &= ~BIT6;                     // Port 1 P1.7 (Indicator #1) Set to off State
82
83      P1DIR |= 0x00;
84      P1OUT &= 0x00;                              // Shut. Down. Everything... :)
85
86      P1REN |= BIT4;                              // Port 1 Resistor enable
87      P1OUT |= BIT4;                              // pull up bit3
88      P1REN |= BIT5;                              // Port 1 Resistor enable
89      P1OUT |= BIT5;                              // pull up bit3
90      __enable_interrupt();
91
92    lcdInit();// Initialize LCD
93
94    //Credits on LCD
95    lcdSetText("Speed Trap 1trak", 0, 0);
96    lcdSetText("Version 1.2 LCD", 0,1);
97    //      ---->________________<---- Maximum 16 Character String Length
98    delay(200);
99    lcdClear();
100   lcdSetText("CC Licensed", 2, 0);
101   lcdSetText("Copyright 2013 ", 1,1);
102   delay(200);
103   lcdClear();
104   lcdSetText("S.D. Hofmeister,", 0, 0);
105   lcdSetText("et al ", 5,1);
106   delay(200);
107   lcdClear();
108   lcdSetText("The Launchpad", 1, 0);
109   lcdSetText("for", 6,1);
110   delay(200);
111   lcdClear();
112   lcdSetText("Model", 5,0);
113   lcdSetText("Railroading", 2,1);
114   delay(200);
115   lcdClear();
116   lcdSetText("Project", 4,0);
117   delay(200);
118   lcdClear();
119   //END Credits on LCD
120
121   lcdSetText("Gate #1 LED Test", 0,0);
122   P1OUT |= BIT0;
123   delay(200);
124   P1OUT &= ~BIT0;
125
126   lcdSetText("Gate #2 LED Test", 0,1);
127   P1OUT |= BIT6;
128   delay(200);
129   P1OUT &= ~BIT6;
130   lcdClear();
131   if ((P1IN & BIT4) == 0) {
132   lcdSetText("IR Gate #1 Pass", 0,0);
133   delay(200);
134   } else {lcdSetText("IR Gate #1 FAIL", 0,0); delay(500);}
135
136   if ((P1IN & BIT5) == 0) {
137   lcdSetText("IR Gate #2 Pass", 0,1);
138   delay(200);
139   } else {lcdSetText("IR Gate #2 FAIL", 0,1); delay(500);}
140   lcdClear();
141
142   while(1)                                    //Loop forever, we work with interrupts!
143     {
144   // Basicallu we loop here doing nothing until an IR Beam is triggered
145
146   lcdSetText("Waiting for", 3, 0);
147   lcdSetText("Train", 6, 1);
148
149             if( (P1IN & BIT4) > 0 && lockgate_2 == 0)   // If Gate #1 is detecting an object
      and Gate #2 is not
150           {
151                 do{
152                     lockgate_1 = 1; // Gate #1 locked so that it will not interrupt
153                    P1OUT |= BIT0; // Turn on RED indication on which Gate #1 or counted
154                     lcdClear();
155                 lcdSetText("Clocking Speed", 1, 0);
156                 lcdSetText(" ", 0,1);
157                 counter_active = 1; // Start Counting from Gate #1 to Gate #2
158                     output_count = 1;    // tell the rest of the program we counted
159                     }while ((P1IN & BIT5) == 0 && lockgate_1 == 1);  // Run the above code
      while Gate #2 is not detecting and Gate #1 is locked
```

```
160           } else {lockgate_1 = 0; P1OUT &= ~BIT0;} // Otherwise unlock Gate #1 and turn off
    RED INDICATOR LED
161
162
163
164            if( (P1IN & BIT5) > 0 && lockgate_1 == 0 ) // If Gate #2 is detecting an object
    and Gate #1 is not
165            {
166            do {
167                 lockgate_2 = 1; // Gate #2 locked so that it will not interrupt
168                 P1OUT |= BIT6; // Turn on GREEN indicator to show Gate #2 as locked
169                 lcdClear();
170                 lcdSetText("Clocking Speed", 1, 0);
171                 lcdSetText(" ", 0,1);
172                 counter_active = 1; // Start Counting from Gate #2 to Gate #1
173                 output_count = 1;  // tell the rest of the program we counted
174                     }while ((P1IN & BIT4) == 0 && lockgate_2 == 1); // Run the above code
    while Gate #1 is not detecting and Gate #2 is locked
175
176               }else {lockgate_2 = 0; P1OUT &= ~BIT6;} // Otherwise unlock Gate #2 and turn
    off GREEN INDICATOR LED
177
178     counter_active = 0;    // done counting - ready to output
179
180           if ( output_count == 1 ) {    // show the count
181                // calculate speed in MPH
182                speed = factor / counter;     // calculate MPH
183                lcdClear();
184                lcdSetText("Train Speed", 2, 0);
185                lcdSetInt(speed, 4, 1);
186                lcdSetText("MPH", 8,1);
187                delay(500);
188                output_count = 0;     // stop displaying and wait for another trigger
189                lcdClear();
190
191       }    // end of IF
192     }      // end of WHILE
193  }    // end of MAIN
194
195  // Timer A0 interrupt service routine
196  #pragma vector=TIMER0_A0_VECTOR
197  __interrupt void Timer_A (void)
198  {
199      if (counter_active == 1) {
200           counter = counter + 1;    // just counting milliseconds
201
202           if (counter > factor) {   // if our counter reaches factor, we're going less than
    1 MPH
203                                     // and we can't display it, so ...
204                counter = 0;         // zero counter
205                counter_active = 0;  // stop counting
206                delay(1000);         // and hang for a bit
207                }
208      }// then return to main
209  }// end of TIMER_A interrupt handler
210
211  // Port 1 interrupt service routine
212  #pragma vector=PORT1_VECTOR
213  __interrupt void Port_1(void)
214  {
215      while (lockgate_1 == 1)  {  // Locks interrupt from disturbing opposite direction
    detection
216          P1IFG &= ~BIT5;                           // P1.4 IFG cleared
217          P1IES ^= BIT5; // toggle the interrupt edge,
218      }
219
220      while (lockgate_2 == 1) { // Locks interrupt from disturbing opposite direction
    detection
221          P1IFG &= ~BIT4;                           // P1.5 IFG cleared
222          P1IES ^= BIT4; // toggle the interrupt edge,
223      }
224          // the interrupt vector will be called
225          // when P1.4 or P1.5 goes from Hitrolow as well as
226          // LowtoHigh
227  }
228
229  void delay(unsigned int ms)   // delay for ms in 10 millisecond intervals
230  {
231   while (ms--)
232      {
233          __delay_cycles(10000); // set for 1 Mhz (10000=10ms)
234      }
235  }
236
```