```
// lab2.c
// Chauncey Yan
// 10.16.14

//  HARDWARE SETUP:
//  PORTA is connected to the segments of the LED display. and to the
    pushbuttons.
//  PORTA.0 corresponds to segment a, PORTA.1 corresponds to segement b, etc.
//  PORTB bits 4-6 go to a,b,c inputs of the 74HC138.
//  PORTB bit 7 goes to the PWM transistor base.

#include <avr/io.h>
#include <util/delay.h>
//#define F_CPU 16000000 // cpu speed in hertz
#define MAX_CHECKS 12 // # checks before a switch is debounced
#define BASE 10  // the base of the clock should be working

uint8_t debounced_state = 0; // Debounced state of the switches
uint8_t state[MAX_CHECKS]; // Array that maintains bounce status
uint8_t id = 0; // Pointer into State
//decimal to 7-segment LED display encodings, logic "0" turns on segment
int dec_to_7seg[18] = {0b11000000, 0b11111001, 0b10100100, 0b10110000, // 0 1 2 3
                       0b10011001, 0b10010010, 0b10000010, 0b11111000, // 4 5 6 7
                       0b10000000, 0b10011000, 0b10001000, 0b10000011, // 8 9 A B
                       0b11000110, 0b10100001, 0b10000110, 0b10001110, // C D E F
                       0b10000011, 0b1000100}; // ^ :

//holds data to be sent to the segments. logic zero turns segment on
int segment_data[5] = {0b11000000, 0xff, 0xff, 0xff, 0xff}; // turn off led not
    needed.


//
    ********************************************************************************
    ****
//                            chk_buttons
//Checks the state of the button number passed to it. It shifts in ones till
//the button is pushed. Function returns a 1 only once per debounced button
//push so a debounce and toggle function can be implemented at the same time.
//Adapted to check all buttons from Ganssel's "Guide to Debouncing"
//Expects active low pushbuttons on PINA port.  Debounce time is determined by
//external loop delay times 12.
void DebounceSwitch(){
    uint8_t i,j;
    state[id++]=0xff - PINA;
    j=0xff;
    for(i=0; i<MAX_CHECKS-1;i++)j=j & state[i];
    debounced_state = j;
    if(id >= MAX_CHECKS)id=0;
}


//
    ********************************************************************************
    *****
//                                  segment_sum
```



```
//takes a 16-bit binary input value and places the appropriate equivalent 4 digit
//BCD segment code in the array segment_data for display.
//array is loaded at exit as:  |digit3|digit2|colon|digit1|digit0|
//
    ********************************************************************************
    *****
void segsum(uint16_t sum) {
    int u = 4;

    //break up decimal sum into 4 digit-segments
    switch (BASE) {
    case 10:
        //breakDgt(sum,10);
        segment_data[0] = dec_to_7seg[sum%10];
        segment_data[1] = dec_to_7seg[sum/10%10];
        segment_data[2] = dec_to_7seg[sum/100%10];
        segment_data[3] = dec_to_7seg[sum/1000%10];
        break;
    case 16:
        breakDgt(sum,4,0x000f);
        break;
    case 8:
        breakDgt(sum,3,0x0007);
        break;
    case 2:
        breakDgt(sum,1,0x0001);
        break;
    }
    //blank out leading zero digits
    while (segment_data[u] == dec_to_7seg[0] || segment_data[u] == 0xff) {
        segment_data[u--] = 0xff;
    }
    //now move data to right place for misplaced colon position
    segment_data[4] = segment_data[3];
    segment_data[3] = segment_data[2];
    segment_data[2] = 0xff; //dec_to_7seg[17];
}

void breakDgt(uint16_t sum, int base_count, uint16_t check_bit) {
    uint16_t dgt;
    int o;
    for (o = 0; o < 4; o++){
        dgt = (sum >> (o * base_count)) & check_bit;
        segment_data[o] = dec_to_7seg[dgt];
    }
}
//
    ********************************************************************************
    ******
// Demostration and debugging functions
//
//
    ********************************************************************************
    ******

void ledDigit (int n, int pos){
```

```
        //pos 0 1 3 4
        if ( pos != -1 ){
            DDRA = 0xff; // output
            PORTB = pos << 4; // selet the digit
            PORTA = dec_to_7seg[n];
        }
        _delay_ms(1);
    }

    void ledNumber (int n, int f){ // f = format
        int digit[4],i,k;
        digit[0] = n%f;
        digit[1] = n/f%f;
        digit[2] = n/f/f%f;
        digit[3] = n/f/f/f%f;
        for (i = 0; i<4; i++){
            k=i%4;
            if (k >= 2) k++;
            ledDigit(digit[i%4],k);
        }
    }

    //
        ********************************************************************************
        ******


    //
        ********************************************************************************
        ******
    int main(){
        int counter = 0,count = 0, j, i, release = 0;
        //set port bits 4-7 B as outputs
        DDRB = 0xf0; // output
        DDRC = 0xff; // output
        PORTC = 0x00;

        DDRF = 0xff;
    while(1){
        //insert loop delay for debounce
        _delay_ms(2);
        //make PORTA an input port with pullups
        PORTA = 0xff;
        DDRA = 0x00;
        //enable tristate buffer for pushbutton switches
        PORTB = 0b01110000;
        _delay_ms(2);


        //now check each button and increment the count as needed
        DebounceSwitch();
        if (debounced_state){ // check if any of the buttons is pressed
            if(release == 0){
                release = 1;
                count += debounced_state;
                //bound the count to 0 - 1023
```



```
                count = count%65536;
                //break up the disp_value to 4, BCD digits in the array: call
                    (segsum)
                segsum(count);
                debounced_state = 0;
            }
        }
        else release = 0;
        //_delay_ms(1);


        //make PORTA an output
        DDRA = 0xff;
        //bound a counter (0-4) to keep track of digit to display
        for (counter = 0; counter < 5; counter++){
            //send PORTB the digit to display
            PORTB = counter << 4;
            //send 7 segment code to LED segments
            PORTA = segment_data[counter];
            //fix for the last digit over bright issue
            if (counter != 4)
                _delay_ms(2);

        }
      }//while
    }//main
```