```c
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include "button7segFunctions.h"

#define TRUE 1
#define FALSE 0
#define true 1
#define false 0
#define True 1
#define False 0

// bits used for digit selection

#define RCLK PB0
#define SCLK PB1
#define MOSI PB2
#define MISO PB3
#define SEL0 PB4
#define SEL1 PB5
#define SEL2 PB6
#define PWM  PB7

// DEMUX to LED wiring
#define SELD1 (0x0 << SEL0)
#define SELD2 (0x1 << SEL0)
#define SELD3 (0x3 << SEL0)
#define SELD4 (0x4 << SEL0)
#define SELDD (0x2 << SEL0)
#define SELBN (0x7 << SEL0)
#define SELCL !SELBN

// Blank 7segment
#define BLNK 0xFF

uint8_t  i;              // for-loop variable

// Holds data to be sent to the segments. logic zero turns segment on
uint8_t segment_data[5];

// Decimal to 7-segment LED display encodings, logic "0" turns on segment
uint8_t dec_to_7seg[12];

// Select digit array
uint8_t digitSelect[8];

// Holds value of buttons from last check
volatile uint8_t buttonState;



//***************************************************************************
//  -- Digit Initialization
//***************************************************************************
void digit_init(){

    // select pins for DEMUX in array form
    digitSelect[0] = SELD1;
    digitSelect[1] = SELD2;
    digitSelect[2] = SELDD;
    digitSelect[3] = SELD3;
    digitSelect[4] = SELD4;
```

```c

    // BCD mapping
    dec_to_7seg[0] = (uint8_t) 0b11000000;
    dec_to_7seg[1] = (uint8_t) 0b11111001;
    dec_to_7seg[2] = (uint8_t) 0b10100100;
    dec_to_7seg[3] = (uint8_t) 0b10110000;
    dec_to_7seg[4] = (uint8_t) 0b10011001;
    dec_to_7seg[5] = (uint8_t) 0b10010010;
    dec_to_7seg[6] = (uint8_t) 0b10000010;
    dec_to_7seg[7] = (uint8_t) 0b11111000;
    dec_to_7seg[8] = (uint8_t) 0b10000000;
    dec_to_7seg[9] = (uint8_t) 0b10010000;
    dec_to_7seg[10] = (uint8_t) 0xFF;

    // 0 is input, 1 is output
    DDRB = (1<<SEL0)|(1<<SEL1)|(1<<SEL2);
    //
    DDRF = (1<<PWM);
    PORTF &= ~(1<<PWM);
}




//**************************************************************************
//    -- 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.
//***************************************************************************
uint8_t chk_button(uint8_t button) {
    // Static array is initialized once at compile time
    static uint16_t State[8] = {0};

    State[button] = (State[button]<<1) | !bit_is_clear(PINA, button) | 0xE000;
    if (State[button] == 0xFF00) return TRUE;
    return FALSE;
} //chk_button




//***************************************************************************
//    -- 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)
{
    //determine how many digits there are
    //break up decimal sum into 4 digit-segments
    //blank out leading zero digits
    //now move data to right place for misplaced colon position

    uint8_t ldZero = TRUE;

    segment_data[0] = sum % 10;
    segment_data[1] = sum/10 % 10;
    segment_data[2] = 10;         // keep colon off; dig10 is mapped to BLNK

    segment_data[3] = sum/100 % 10;
    segment_data[4] = sum/1000 % 10;

    // Covert dec to BCD, ignoring colon and blanking leading zeros
    //ldZero=TRUE -> index has not yet found a non-zero digit
    for (i=4; i > 0; --i)
    {
        if (ldZero && (segment_data[i]==0))
            segment_data[i] = BLNK;
        else
        {
            if (i!=2) ldZero = FALSE;
            segment_data[i] = dec_to_7seg[segment_data[i]];
        }//if
    }//for

    segment_data[0] = dec_to_7seg[segment_data[i]];

    return;
}//segment_sum




//*******************************************************************************
//    -- Checks State of Buttons on 7seg Bus --
//*******************************************************************************
void toggle_button_bus() {

    //make PORTA an input port with pullups
    DDRA = 0x00;     // 0 is input, 1 is output
    PORTA = 0xFF;    // 0 is float, 1 is pull-up

    //enable tristate buffer for pushbutton switches
    PORTB &= SELCL;
    PORTB |= SELBN;

    //buttonState=0;
    int i;
    //now check each button and increment the count as needed
    for (i=0; i<8; i++)
    {
        if (chk_button(i))
            buttonState ^= 1<<i;
    }//for


    //disable tristate buffer for pushbutton switches
    PORTB &= SELCL;

    // Reset A as output
    DDRA = 0xFF;
}
```