```
/* ========================================
 *
 * Copyright YOUR COMPANY, THE YEAR
 * All Rights Reserved
 * UNPUBLISHED, LICENSED SOFTWARE.
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION
 * WHICH IS THE PROPERTY OF your company.
 *
 * ========================================
*/

/*  ================================================
    ====== PROJECT: ULTRASONIC DISTANCE METER ======
    ========== UNIT: TRC3500 | GROUP: C9 ===========
    ====== ID: 27952827 | NAME: JEHAD BAYAZID ======
    ========== LAST MODIFIED: 13/06/2020 ===========
    ================================================
*/
#include "project.h"
#include <stdio.h>
#include <stdlib.h>

int digit, mode, group_number, unit, time, distance, timer_on, offset_val;
int n, i, dist[10], counter = 0;
int Btn0, Btn1, Btn2, Btn3, Btn4 = 0;


// -------------------------------
/* ===== BUTTON INTERRUPTS ===== */
// -------------------------------
CY_ISR(Button_0)        // Distance Measurement Mode
{
    Btn0 = 1;
}
CY_ISR(Button_1)        // Unit Selection Mode
{
    Btn1 = 1;
}
CY_ISR(Button_2)        // Decrease
{
    Btn2 = 1;
}
CY_ISR(Button_3)        // Increase
{
    Btn3 = 1;
}
CY_ISR(Button_4)        // Program Mode
{
    Btn4 = 1;
    Btn2 = 0;
    Btn3 = 0;
}
```

```
CY_ISR(Switch_4)        // One-second button trigger
{
    Counter_Switch_Start();
}
CY_ISR(Distance_Timer)  // Distance Timer Interrupt
{
    Timer_UDM_Stop();
    timer_on = 0;
}

/* --- FUNCTION: Sleep mode --- */
    void LED_Sleep_Mode()
    {
        LED_Driver_PutDecimalPoint(0,3);    // remove decimal point
        CyDelay(1000);
        LED_Driver_PutDecimalPoint(1,3);    // put decimal point
        CyDelay(1000);
    }

/* --- FUNCTION: Buzzer trigger --- */
    void Buzzer()
    {
        buzzer_Write(0);
        CyDelay(200);
        buzzer_Write(1);
    }

/* --- FUNCTION: Calculate distance --- */
    void Find_Distance(int time)
    {
        distance = 0.5*time*28.412;
    }

/* --- FUNCTION: Display distance --- */
    void Display_Measurement(int distance)
    {
        if ( unit == 105 ){             // if selected unit is inches
            distance = distance/2.54;   // converts to inches
            Unit_Sel_Write(1);          // turn on built-in LED to indicate ↵
unit
        }
        LED_Driver_Write7SegNumberDec(distance, 0, 4, LED_Driver_RIGHT_ALIGN);
        LED_Driver_PutDecimalPoint(1,1);    // add decimal point
        CyDelay(2000);                      // keep display for 2 seconds
        Unit_Sel_Write(0);
        LED_Driver_ClearDisplayAll();
    }

/* --- FUNCTION: Reset Distance Measurement --- */
    void Reset_UDM()
    {
        Timer_Reset_Write(1);
        Transmit_Write(0);
```

```
        time = 0;
        distance = 0;

        Opamp_1_Stop();
        PGA_1_Stop();
        Comp_1_Stop();
        VDAC8_1_Stop();
        LED_Write(0);
    }

/* --- FUNCTION: Calibration table --- */
    int Find_Offset()
    {
        if      ( distance <= 299 )  { return 0;   }    // invalid measurement
        else if ( distance <= 399 )  { return 30;  }    // 2 - 3 cm
        else if ( distance <= 499 )  { return 292; }    // 2 cm
        else if ( distance <= 550 )  { return 278; }    // 2.5 cm
        else if ( distance <= 640 )  { return 271; }    // 3 - 3.5 cm
        else if ( distance <= 699 )  { return 260; }    // 4 cm
        else if ( distance <= 810 )  { return 245; }    // 5 - 5.5 cm
        else if ( distance <= 895 )  { return 240; }    // 6 - 6.5 cm
        else if ( distance <= 1020 ) { return 235; }    // 7 - 7.5 cm
        else if ( distance <= 1045 ) { return 220; }    // 8 cm
        else if ( distance <= 1075 ) { return 250; }    // 8.5 cm
        else if ( distance <= 1110 ) { return 265; }    // 9 - 9.5 cm
        else if ( distance <= 1220 ) { return 230; }    // 10 - 10.5 cm
        else if ( distance <= 1389 ) { return 200; }    // 11 cm
        else if ( distance <= 1425 ) { return 200; }    // 11.5 cm
        else if ( distance <= 1455 ) { return 195; }    // 12 cm
        else if ( distance <= 1499 ) { return 185; }    // 12.5 cm
        else if ( distance <= 1535 ) { return 170; }    // 13.5 cm
        else if ( distance <= 1630 ) { return 125; }    // 14 cm
        else if ( distance <= 1735 ) { return 105; }    // 14.5 cm
        else if ( distance <= 1800 ) { return 60; }     // 15.5 cm

        else { return 50; }                             // 16 - 30 cm
    }

/* --- FUNCTION: Array sorting --- */
    int cmpfunc (const void * a, const void * b) {
        return ( *(int*)a - *(int*)b );
    }

/* --- FUNCTION: Button & mode reset  --- */
    void Reset_Buttons()
    {
        Btn0 = 0;
        Btn1 = 0;
        Btn2 = 0;
        Btn3 = 0;
        Btn4 = 0;
    }
```

```
/* --- FUNCTION: Mode toggle --- */
    int Mode()
    {
        if (Btn0){                  // Measurement Mode
            return 1;
        }
        else if (Btn1){             // Unit Selection Mode
            return 2;
        }
        else if (Btn4){             // Program Mode
            return 3;
        }
        else if (Btn2 & ~Btn3){     // Increment | Inches
            return 4;
        }
        else if (~Btn2 & Btn3){     // Decrement | Centimeters
            return 5;
        }
        else {                      // Sleep Mode
            return 0;
        }
    }

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */

    // Start Modules
    LED_Driver_Start();         // 7-Segment Display
    EEPROM_Start();             // memory module

    // Starting Interrupts
    isr_0_ClearPending(), isr_1_ClearPending(), isr_2_ClearPending();
    isr_3_ClearPending(), isr_4_ClearPending(), isr_timer_ClearPending();
    isr_switch_ClearPending();

    isr_0_StartEx(Button_0), isr_1_StartEx(Button_1);
    isr_4_StartEx(Button_4), isr_timer_StartEx(Distance_Timer);
    isr_switch_StartEx(Switch_4);


    // 7-Segment Display Startup
    for ( digit = 0; digit < 4; digit++ )
    {
        LED_Driver_Write7SegDigitDec(8,digit);
        LED_Driver_PutDecimalPoint(1,digit);
        CyDelay(1000);

        LED_Driver_ClearDisplay(digit);
    }

    // Display group number
    LED_Driver_Write7SegDigitHex(0x0Cu,2);
```

```
    LED_Driver_Write7SegDigitDec(EEPROM_ReadByte(0), 3);

    CyDelay(1000);
    LED_Driver_ClearDisplayAll();

    // Display last-set unit
    unit = EEPROM_ReadByte(1);
    LED_Driver_PutChar7Seg(unit, 3);
    CyDelay(1000);
    LED_Driver_ClearDisplayAll();


    for(;;)
    {
        Mode ();         // updating mode

        //  ===== SLEEP MODE =====
        while ( Mode() == 0 ) {
            LED_Sleep_Mode();
        }

        switch ( Mode() ) {

            //  ===== MEASUREMENT MODE =====
            case 1:
            Buzzer();

            for (i = 0; i < 10; i++){       // taking measurements 10 times
                Opamp_1_Start();
                PGA_1_Start();
                Comp_1_Start();
                VDAC8_1_Start();

                timer_on = 1;
                Timer_Reset_Write(0);
                Transmit_Write(1);
                Timer_UDM_Start();

                while (timer_on) {      // waiting unit sent signal is received
                CyDelay(10);
                }
                time = Timer_UDM_ReadCounter(); // read time taken from timer
                Find_Distance(65536-time);      // subtracting timer period
                dist[i] = distance;             // pushing found distances ↵
into array

                if (distance < 200)             // repeat measurement for ↵
invalid distances
                { i--; }

                Reset_UDM();                    // resetting for next iteration
                CyDelay(50);
            }
```

```
            qsort(dist, 10, sizeof(int), cmpfunc);      // sorting calculated ↵
distances

            distance = (dist[4]+dist[5]+dist[6]+dist[7])/4; // avg from ↵
middle of array
            Display_Measurement(distance - Find_Offset());  // minus offset ↵
from distance

            Reset_UDM();
            Reset_Buttons();        // exit mode

            break;

            //  ===== UNIT SELECTION MODE =====
            case 2:
            Buzzer();
            LED_Driver_ClearDisplayAll();
            Btn1 = 0;
            isr_2_StartEx(Button_2), isr_3_StartEx(Button_3);

            LED_Driver_PutChar7Seg(105, 0);     // ASCI i for inches
            LED_Driver_PutChar7Seg(99, 3);      // ASCI c for centimetres
            while ( Mode() != 2 ) {
                if (Btn2) {
                    LED_Driver_ClearDisplay(3);
                    LED_Driver_PutChar7Seg(105, 0);     // for inches

                    Unit_Sel_Write(1);      // turn on inches LED
                    unit = 105;             // assign i to unit variable

                    Btn2 = 0;
                }
                if (Btn3) {
                    LED_Driver_ClearDisplay(0);
                    LED_Driver_PutChar7Seg(99, 3);     // for centimetres

                    Unit_Sel_Write(0);      // turn off inches LED
                    unit = 99;              // assign c to unit variable

                    Btn3 = 0;
                }
            }

            EEPROM_WriteByte( (unit), 1);   // save unit in memory
            Reset_Buttons();                // exit mode
            Unit_Sel_Write(0);              // turn off LED after selection

            LED_Driver_ClearDisplayAll();

            break;

            //  ===== PROGRAM MODE =====
            case 3:
```

```
            Buzzer();

            Btn2 = 0, Btn3 = 0;
            while ( Mode() == 3 ) {
                group_number = EEPROM_ReadByte(0);              // read saved ↵
group no.
                Btn2 = 0, Btn3 = 0;

                LED_Driver_Write7SegDigitDec(group_number, 3); // display ↵
group no.
                CyDelay(1000);
                LED_Driver_ClearDisplay(3);
                CyDelay(500);

                if ( Btn2 && (group_number > 0) ) {          // decreament
                    EEPROM_WriteByte( (group_number-1), 0);

                    Btn2 = 0;
                }
                if ( Btn3 && (group_number < 9) ) {          // increament
                    EEPROM_WriteByte( (group_number+1), 0);

                    Btn3 = 0;
                }
                if (Btn1) {          // Exit to unit selection mode
                    Btn2 = 0;
                    Btn3 = 0;
                    Btn4 = 0;
                    isr_2_Stop(), isr_3_Stop();
                }
                if (Btn0) {          // Exit to sleep mode
                    Reset_Buttons();
                    isr_2_Stop(), isr_3_Stop();
                }

                isr_2_StartEx(Button_2), isr_3_StartEx(Button_3);
            }
            break;

            default:
            Reset_Buttons();        // exit to sleep mode

            break;
        }

    }

}

/* [] END OF FILE */
```