```c
/*****************************************************************************************
* Lab3.c - Function generator designed with uC/OS. User can set amplitude, frequency,
*          and type of wave via the K65 Tower Board's touch sensing electrodes and keypad.
*
* 02/01/2018, Created project. Brian Willis
* 02/07/2018, Rod Mesecar added to project
* 02/07/2018, Anthony Needles added to project
*****************************************************************************************/
#include "MCUType.h"
#include "app_cfg.h"
#include "os.h"
#include "K65TWR_GPIO.h"
#include "uCOSKey.h"
#include "LcdLayered.h"
#include "DMA.h"
#include "TSI.h"
#include "Wave.h"

/*****************************************************************************************
* Defined Constants
*****************************************************************************************/
#define CURSORSTART 10
#define UI_TASK_MSG_Q_SIZE 5

/*****************************************************************************************
* Allocate task control blocks
*****************************************************************************************/
static OS_TCB AppTaskStartTCB;
static OS_TCB UITaskTCB;
static OS_TCB UITSISrvTaskTCB;
static OS_TCB UIKeySrvTaskTCB;

/*****************************************************************************************
* Allocate task stack space
*****************************************************************************************/
static CPU_STK AppTaskStartStk[APP_CFG_TASK_START_STK_SIZE];
static CPU_STK UITaskStk[APP_CFG_UI_TASK_STK_SIZE];
static CPU_STK UITSISrvTaskStk[APP_CFG_UITSISRV_TASK_STK_SIZE];
static CPU_STK UIKeySrvTaskStk[APP_CFG_UIKEYSRV_TASK_STK_SIZE];

/*****************************************************************************************
* Task Function Prototypes
*****************************************************************************************/
static void AppStartTask(void *p_arg);
static void UITask(void *p_arg);
static void UITSISrvTask(void *p_arg);
static void UIKeySrvTask(void *p_arg);

/*****************************************************************************************
* Private Resources
*****************************************************************************************/
static WAVE_W dispWave;         //Local wave that displays waveform from Wave.c
static WAVE_W setWave;          //Local wave that gets adjusted by UI Task
static INT8U cursorLoc = CURSORSTART;

/*****************************************************************************************
* main()
*****************************************************************************************/
void main(void){
    OS_ERR os_err;
    CPU_IntDis();                                                //Disable all interrupts, OS will enable them
    OSInit(&os_err);                                             //Initialize uC/OS-III
    while(os_err != OS_ERR_NONE){}                               //Error Trap

    OSTaskCreate(&AppTaskStartTCB,                               //Address of TCB assigned to task
                 "Start Task",                                   //Name you want to give the task
                 AppStartTask,                                   //Address of the task itself
                 (void *) 0,                                     //p_arg is not used so null ptr
                 APP_CFG_TASK_START_PRIO,                        //Priority you assign to the task
                 &AppTaskStartStk[0],                            //Base address of task’s stack
                 (APP_CFG_TASK_START_STK_SIZE/10u),              //Watermark limit for stack growth
                 APP_CFG_TASK_START_STK_SIZE,                    //Stack size
                 0,                                              //Size of task message queue
                 0,                                              //Time quanta for round robin
                 (void *) 0,                                     //Extension pointer is not used
                 (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),    //Options
                 &os_err);                                       //Ptr to error code destination
    while(os_err != OS_ERR_NONE){}                               //Error Trap

    OSStart(&os_err);                                            //Start multitasking (i.e. give control to uC/OS)
    while(os_err != OS_ERR_NONE){}                               //Error Trap
}

/*****************************************************************************************
* This should run once and be suspended. Could restart everything by resuming.
* (Resuming not tested)
* Todd Morton, 01/06/2016
* Modified for Lab 3: Brian Willis, 02/16/2018
*****************************************************************************************/
static void AppStartTask(void *p_arg) {
    OS_ERR os_err;
    (void)p_arg;                                //Avoid compiler warning for unused variable
    OS_CPU_SysTickInitFreq(DEFAULT_SYSTEM_CLOCK);

    /* Initialize StatTask. This must be called when there is only one task running.
     * Therefore, any function call that creates a new task must come after this line.
     * Or, alternatively, you can comment out this line, or remove it. If you do, you
     * will not have accurate CPU load information                                       */
//    OSStatTaskCPUUsageInit(&os_err);

    //Initialize peripherals
    LcdInit();
    KeyInit();
    TSIInit();
```

```c
     GpioDBugBitsInit();
     DMAInit(*DMAWavCurSamples);
     DMADAC0Init();
     DMAPIT0Init();
     WaveInit();

     WaveGet(&setWave);                          //Initialize local waves
     WaveGet(&dispWave);

     LcdDispClear(WAVE_LAYER);

     OSTaskCreate(&UITaskTCB,                    //Create UITask
                  "UI Task",
                  UITask,
                  (void *) 0,
                  APP_CFG_UI_TASK_PRIO,
                  &UITaskStk[0],
                  (APP_CFG_UI_TASK_STK_SIZE / 10u),
                  APP_CFG_UI_TASK_STK_SIZE,
                  UI_TASK_MSG_Q_SIZE, // Task Msg Queue
                  0,
                  (void *) 0,
                  (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
                  &os_err);
    while(os_err != OS_ERR_NONE){}              //Error Trap

    OSTaskCreate(&UITSISrvTaskTCB,              //Create UITSISrvTask
                    "UI Touch Service Task",
                    UITSISrvTask,
                    (void *) 0,
                    APP_CFG_UITSISRV_TASK_PRIO,
                    &UITSISrvTaskStk[0],
                    (APP_CFG_UITSISRV_TASK_STK_SIZE / 10u),
                    APP_CFG_UITSISRV_TASK_STK_SIZE,
                    0,
                    0,
                    (void *) 0,
                    (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
                    &os_err);
     while(os_err != OS_ERR_NONE){}              //Error Trap

     OSTaskCreate(&UIKeySrvTaskTCB,              //Create UITSISrvTask
                  "UI Key Service Task",
                  UIKeySrvTask,
                  (void *) 0,
                  APP_CFG_UIKEYSRV_TASK_PRIO,
                  &UIKeySrvTaskStk[0],
                  (APP_CFG_UIKEYSRV_TASK_STK_SIZE / 10u),
                  APP_CFG_UIKEYSRV_TASK_STK_SIZE,
                  0,
                  0,
                  (void *) 0,
                  (OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR),
                  &os_err);
     while(os_err != OS_ERR_NONE){}              //Error Trap

     OSTaskSuspend((OS_TCB *)0, &os_err);
     while(os_err != OS_ERR_NONE){}              //Error Trap
 }

 /*****************************************************************************************
 * UITask() - Controls the user interface. Displays current waveform to LCD and
 *             updates waveform based on user inputs.
 *
 * 02/16/2018, Brian Willis
 *****************************************************************************************/
 static void UITask(void *p_arg){
     OS_ERR os_err;
     INT8U *msgp;
     OS_MSG_SIZE msg_size;
     (void)p_arg;

     while(1){
         (void)LcdCursor(2, cursorLoc, WAVE_LAYER, TRUE, TRUE);  //Display cursor
         WaveGet(&dispWave);                                     //Get current waveform to display its properties

         //Display current waveform to LCD
         LcdDispDecByte(1, 2, WAVE_LAYER, dispWave.amp, 1);
         LcdDispString(1, 1, WAVE_LAYER, "A:");
         //Display lower 2 chars of frequency
         LcdDispDecByte(1, CURSORSTART+2, WAVE_LAYER, (INT8U)(dispWave.freq-((dispWave.freq/100)*100)), 1);
         //Middle 2 chars
         LcdDispDecByte(1, CURSORSTART, WAVE_LAYER, (INT8U)(((dispWave.freq-((dispWave.freq/10000)*10000))
                                                             -(dispWave.freq-((dispWave.freq/100)*100)))/100), 1);
         //Upper char
         LcdDispDecByte(1, CURSORSTART-2, WAVE_LAYER, (INT8U)((dispWave.freq)/10000), 1);
         LcdDispString(1, 8, WAVE_LAYER, "F:");
         LcdDispString(1, 15, WAVE_LAYER, "Hz");
         if(dispWave.waveshape == SIN){
             LcdDispString(2, 1, WAVE_LAYER, "SINE");
         } else{
             LcdDispString(2, 1, WAVE_LAYER, "TRI ");
         }

         //Frequency being changed by user
         LcdDispDecByte(2, CURSORSTART+2, WAVE_LAYER, (INT8U)(setWave.freq-((setWave.freq/100)*100)), 1);
         LcdDispDecByte(2, CURSORSTART, WAVE_LAYER, (INT8U)(((setWave.freq-((setWave.freq/10000)*10000))
                                                             -(setWave.freq-((setWave.freq/100)*100)))/100), 1);
         LcdDispDecByte(2, CURSORSTART-2, WAVE_LAYER, (INT8U)((setWave.freq)/10000), 1);
         LcdDispString(2, 6, WAVE_LAYER, "Set:");
         LcdDispString(2, 15, WAVE_LAYER, "Hz");

         //Turn off debug bit while waiting
         DB4_TURN_OFF();
```

```c
        //Wait for either key press or TSI
        msgp = OSTaskQPend(0, OS_OPT_PEND_BLOCKING, &msg_size, (CPU_TS *)0, &os_err);
        //Error Trap
        while(os_err != OS_ERR_NONE){}
        //Turn on debug bit while ready/running
        DB4_TURN_ON();

        switch(*msgp){
            case(1):                //Left electrode touched
                if(setWave.amp < 20){
                    setWave.amp++;
                    WaveSet(&setWave, 1);
                } else{}
                break;
            case(2):                //Right electrode touched
                if(setWave.amp > 0){
                    setWave.amp--;
                    WaveSet(&setWave, 1);
                } else{}
                break;
            default:                //Key press
                if(*msgp == 0x11){                          //'A' - Set to sinewave
                    setWave.waveshape = SIN;
                    WaveSet(&setWave, 3);
                } else if(*msgp == 0x12){                   //'B' - Set to triangle wave
                    setWave.waveshape = TRI;
                    WaveSet(&setWave, 3);
                } else if((*msgp >= 48) && (*msgp <= 57)){  //0-9
                    *msgp = *msgp - 48;                     //Convert ASCII to decimal
                    switch(cursorLoc){                      //Current location of cursor determines digit to update
                        case(CURSORSTART):                  //Update 10,000s place
                            setWave.freq = ((*msgp)*10000)+(setWave.freq-((setWave.freq/10000)*10000));
                            cursorLoc++;
                            break;
                        case(CURSORSTART+1):                //1,000s
                            setWave.freq =
                            *msgp*1000+((setWave.freq/10000)*10000)+(setWave.freq-((setWave.freq/1000)*1000));
                            cursorLoc++;
                            break;
                        case(CURSORSTART+2):                //100s
                            setWave.freq =
                            *msgp*100+((setWave.freq/1000)*1000)+(setWave.freq-((setWave.freq/100)*100));
                            cursorLoc++;
                            break;
                        case(CURSORSTART+3):                //10s
                            setWave.freq = *msgp*10+((setWave.freq/100)*100)+(setWave.freq-((setWave.freq/10)*10));
                            cursorLoc++;
                            break;
                        case(CURSORSTART+4):                //1s
                            setWave.freq = *msgp+((setWave.freq/10)*10);
                            break;
                    }
                } else if(*msgp == 0x14){                       //'D' - Backspace
                    if(cursorLoc > CURSORSTART){
                        cursorLoc--;
                    } else{}
                } else if(*msgp == 0x23){                       //'#' - Enter
                    if(setWave.freq > 10000){
                        setWave.freq = 10000;
                    } else if(setWave.freq < 10){
                        setWave.freq = 10;
                    } else{
                        WaveSet(&setWave, 2);
                        dispWave = setWave;
                    }
                    cursorLoc = CURSORSTART;
                } else{}
                break;
        }
    }
}

/*****************************************************************************************
* UITSISrvTask() - Pends on TouchPend and updates UIInputQ. Gives TSI data to UITask().
*
* 02/09/2018, Rod Mesecar
* 02/16/2018, Modified for UITask(). Brian Willis
*****************************************************************************************/
static void UITSISrvTask(void *p_arg){
    OS_ERR os_err;
    (void)p_arg;
    INT8U tsipress;
    OS_MSG_SIZE msg_size = sizeof(tsipress);

    while(1){
        DB5_TURN_OFF();
        tsipress = TSITouchPend(&os_err);
        while(os_err != OS_ERR_NONE){}      //Error Trap
        DB5_TURN_OFF();

        OSTaskQPost(&UITaskTCB, &tsipress, msg_size, OS_OPT_POST_FIFO, &os_err);    //Place TSI data into queue
        while(os_err != OS_ERR_NONE){}      //Error Trap
    }
}

/*****************************************************************************************
* UIKeySrvTask() - Pends on KeyPend and updates UIInputQ. Gives keypad data to UITask().
*
* 02/09/2018, Rod Mesecar
* 02/16/2018, Modified for UITask(). Brian Willis
*****************************************************************************************/
static void UIKeySrvTask(void *p_arg){
    OS_ERR os_err;
    (void)p_arg;
    INT8U keypress = 0;
    OS_MSG_SIZE msg_size = sizeof(keypress);

    while(1){
        DB6_TURN_OFF();
        keypress = KeyPend(0, &os_err);      //Wait for key press
        while(os_err != OS_ERR_NONE){}       //Error Trap
        DB6_TURN_OFF();

        OSTaskQPost(&UITaskTCB, &keypress, msg_size, OS_OPT_POST_FIFO, &os_err);    //Place keypress into queue
        while(os_err != OS_ERR_NONE){}       //Error Trap
    }
}
```