```
/*******************
Mark Moerdyk
First modification: 2/6/13
Last modification: 2/12/13
************************/

#include "includes.h"

#define START_ADDR (INT8U *)0xC000
#define END_ADDR (INT8U *)0xFFFF
#define SLICE_PER 10
#define DC1 (INT8U)0x11
#define DC2 (INT8U)0x12
#define DC3 (INT8U)0x13
#define DC4 (INT8U)0x14


/*********************************************************************
* Public Event Definitions
*********************************************************************/
OS_EVENT *SecFlag;                /* A one second flag semaphore       */

/*********************************************************************
* Task Function Prototypes.
*   - Private if in the same module as startup task. Otherwise public.
*********************************************************************/
static void  StartTask(void *p_arg);
static void  DemoCntrlTask(void *p_arg);
static void  LCDDemoTask(void *p_arg);
void DisplayCheckSum(INT16U TotalCheckSum);
INT16U CalcChkSum( INT8U *startaddr, INT8U *endaddr);

/*********************************************************************
* Allocate task stack space.
*********************************************************************/
OS_STK  StartTaskStk[STARTTASK_STK_SIZE];
OS_STK  DemoCntrlTasklStk[DEMOCNTRLTASK_STK_SIZE];
OS_STK  LCDDemoTaskStk[LCDDEMOTASK_STK_SIZE];

/*********************************************************************
* main()
Includes: Initialize OS, Key, and LCD
Creates start task
*********************************************************************/
void main(void)
{

    DBUG_PORT = 0x00;      //Initialize Debug bits
    DBUG_PORT_DIR = DB_OUTS;


    OSInit();                         /* Initialize uC/OS-II             */
    KeyInit();
    LcdInit();

    (void)OSTaskCreate(StartTask,         /* Create Startup Task         */
               (void *)0,
               (void *)&StartTaskStk[STARTTASK_STK_SIZE],
               STARTTASK_PRIO);

    SecFlag = OSSemCreate(0);       /* Create a semaphore flag           */

    OSStart();                        /* Start multitasking              */
}

/*********************************************************************
* STARTUP TASK - Prints out checksum and waits for c press. When C is pressed,
* starts LCD and Demo Task, then deletes itself
* Functions included: CalcChkSum, LcdDispStrg, DisplayCheckSum
* Creates: LCDDemoTask and DemoCntrlTask
*********************************************************************/
static void StartTask(void *p_arg)
{
    INT16U calcchksumresult;
    INT8U keypress = 0;
    INT16U key;
    INT8U err;

    (void)p_arg;                           /* Avoid compiler warning     */
    OSTickInit();
    LcdClrDisp();
    LcdMoveCursor(2,1);

    DBUG_PORT |= PP7;



    calcchksumresult = CalcChkSum( START_ADDR, END_ADDR);

    LcdDispStrg("CS: "); //Prints array based string

    DisplayCheckSum(calcchksumresult);

    while( keypress != DC3)
    {
      keypress = KeyPend(key, &err);
    }

    (void)OSTaskCreate(LCDDemoTask,         /* Create LCD Demo Task              */
               (void *)0,
               (void *)&LCDDemoTaskStk[LCDDEMOTASK_STK_SIZE],
               LCDDEMOTASK_PRIO);

    (void)OSTaskCreate(DemoCntrlTask,         /* Create Demo Control Task              */
               (void *)0,
               (void *)&DemoCntrlTasklStk[DEMOCNTRLTASK_STK_SIZE],
               DEMOCNTRLTASK_PRIO);


    DBUG_PORT &= ~PP7;
    (void)OSTaskDel(STARTTASK_PRIO);
    FOREVER()
    {
      //do nothing
    }


}
/*********************************
*CalcChkSum Function - A function that takes
the contents of the start and end address and adds the
contents from the start address to the end address

Passes in: startaddr, endaddr
Returns: TotalSum
```

```
************************************/
INT16U CalcChkSum( INT8U *startaddr, INT8U *endaddr)
{
    INT16U totalsum = 0x0000;
    INT8U *add = startaddr;

    while(add != endaddr)//includes endaddress content
    {
        totalsum = (INT16U)*add + totalsum; // adds the 16bit content to totalsum
        add++;

    }
    totalsum = (INT16U)*endaddr + totalsum;
    return totalsum;


}
/***************************
*DisplayCheckSum - Function that takes the 16 bit
sum and displays it into two 8 bit bytes on the LCD

*Modules: LCD
*Member: LcdDispByte()
***************************/
void DisplayCheckSum(INT16U TotalCheckSum)
{
    INT8U highbit;
    INT8U lowbit;

    highbit = ((INT8U)(TotalCheckSum >> 8)); //high 8 bits of the 16 bit interger
    LcdDispByte(&highbit);
    lowbit =(INT8U) TotalCheckSum; //default to the low 8 bit of the 16 bits
    LcdDispByte(&lowbit);

}

/*********************************************************************
* DemoCntrlTask - Waits for the pressing of B on the key pad. When B is
pressed, then LCDDemoTask is deleted and restarted.
Creates: LCDDemoTask
Deletes: LCDDemoTask
Uses: OSTimeDly, KeyPend
*********************************************************************/

static void DemoCntrlTask(void *p_arg)
{
   INT8U secondkeypress = 0;
   INT16U secondkey;
   INT8U error;

   (void)p_arg;
   FOREVER()
   {
        DBUG_PORT |= PP4;
        OSTimeDly(10);
        DBUG_PORT &= ~PP4;

        secondkeypress = KeyPend(secondkey, &error);

        if( secondkeypress == DC2)
        {
            (void)OSTaskDel(LCDDEMOTASK_PRIO); // deletes LCDDemoTask
            (void)OSTaskCreate(LCDDemoTask,      /* Creates LCDDemoTask */
                (void *)0,
                (void *)&LCDDemoTaskStk[LCDDEMOTASK_STK_SIZE],
                LCDDEMOTASK_PRIO);

        }
        else
        {
            //do nothing
        }

   }
}

/*********************************************************************
* LCDDemoTask - Runs through the LCD demo task of lab 2, and displays the
results on the LCD display
Uses: LCD module
Also uses: OSTimeDly
*********************************************************************/
static void LCDDemoTask(void *p_arg)
{
    INT8U dispbytenum = 0x09;
    INT8U delaynumber = 150;
    (void)p_arg;
    LcdClrDisp();

    FOREVER()
    {

        DBUG_PORT |= PP5;
        LcdDispChar('B');
        LcdCursor(FALSE,FALSE);
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
        DBUG_PORT |= PP5;
        LcdDispTime ( 4,45,8);
        LcdCursor(FALSE, TRUE);
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
        DBUG_PORT |= PP5;
        LcdClrLine(1);
        LcdMoveCursor(2,1);
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
        DBUG_PORT |= PP5;
        LcdDispDecByte(&dispbytenum,TRUE);
        LcdCursor(TRUE, FALSE);
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
        DBUG_PORT |= PP5;
        LcdClrLine(2);
        LcdFSpace();
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
        DBUG_PORT |= PP5;
        LcdDispDecByte(&dispbytenum, FALSE);
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
        DBUG_PORT |= PP5;
        LcdCursor(TRUE, TRUE);
        LcdBSpace();
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
```

```
        DBUG_PORT |= PP5;
        LcdBSpace();
        LcdDispChar('F');
        DBUG_PORT &= ~PP5;
        OSTimeDly(2000);
        DBUG_PORT |= PP5;
        LcdClrDisp();


    }

}
```