

Use of semaphores and multitasking to
Implement a user interfaced temperature simulator
using MicroC/OS-II RTOS

LAB #7 EE 2160
November 15, 2004

BY:
Narayanan Krishnamurthy

TABLE OF CONTENTS

TABLE OF CONTENTS.....	2
Purpose:.....	3
Procedure:	3
Design:	4
State Diagrams:.....	5
ISR Handler –state diagram	5
Temperature simulator – state diagram.....	5
Temperature simulator – state diagram.....	6
Pseudo-Code:	6
Bubble sort – pseudo code	6
Flow_Chart For User Menu Display.....	7
Flow-Chart For Avg Temp Display	7
Implementation:	8
//lab7.h –Header file.....	8
//defines used by user interface FSM.....	8
//defines used in ISR for decoding digit	8
//defines used temperature setting simulator.....	9
//multitask.c file:	9
//ten second –system time task –priority 1.....	10
//temperature-simulator task –priority 2	11
//temp setting view task –avg of latest four settings –priority 3	12
//user interface -menu display task –priority 4	12
// background -bubble sorting task –priority 5.....	13
// bubbleSort and its associated routines.....	14
//ISR handler –decodes digit & cntrl’s state transitions of user interface.....	15
//decodeDigit routine called by ISR handler.....	18
//main module – task & semaphore creation, ISR,LCD & OS initialization.....	19
//OSTimeTickHook module in os_cpu_c.c –nr_timer1 based scheduling	20
Validation and Testing:.....	21
Results:.....	22
Conclusion:	25

Purpose:

The purpose of lab 7 was to understand RTOS functions of multitasking, process synchronization and scheduling. By building a simulator for the user interface built in the earlier lab, we understand the resourcefulness of the OS in gluing the different elements designed independently, and ultimately integrating the individual processes into the complex multitasking simulator.

Procedure:

In Lab 7 the temperature simulator module was written to increment a seed temp setting value by two degrees, the temp setting values were stored in a circular array, with the array index incremented under a modulo operation (i.e. array index reset to zero upon reaching the length of the array). A timer1 Avalon timer interval was created in the CPU, the timer acts as the overall tick based timer for the ucos RTOS. The supplemental handout was used to test the functionality of the timer1 based scheduling of tasks.

Multitasking and task synchronization, signaling using semaphores enabled us to integrate individual modules with minimal changes to support intertask communication/signaling.

The various tasks that were identified for the temperature simulator for lab 7 are:

- User input ISR – asynchronous highest priority
- Ten second system time display – priority 1
- Temperature simulator –priority 2
- Average temp calculation and display of latest four temp settings –priority 3
- User interface menu –priority 4
- Bubble sort task – lowest priority.

The individual tasks were tested before integration, lab7.h has all the defines and multitask.c has all the tasks integrated in it.

Design:

The bubble sorting task generates 1000 random numbers and sorts them in the background as the lowest priority task. These 1000 numbers were generated using the `rand ()` and `srand ()` built-in functions. The random numbers were also modifiable by changing the seed value. Once the 1000 numbers were generated, they were stored in an array. A value of 1000 was chosen instead of the proposed 100,000 value due to limitation of stack size. Once the task completes sorting it displays the first ten numbers of the sorted array and the task is deleted from the scheduler using `OSTaskDel (OS_PRIO_SELF)` command of ucOS RTOS.

The temperature simulator was initialized by a hardcoded value for the seed, and was checked for an increment of two for every ten seconds, once a ten degree difference was reached the temperature generated by the simulator wraps back to the seed value.

The user interface task and ISR handler handles the user input to view / change temperature settings. The view temperature function signals the average temperature task to calculate the average temperature setting of the latest four readings. Note: array index has to be manipulated in a circular manner to get to the latest four temp settings of the circular array.

The Set function of the user interface stores the new temp setting in the seed and in the circular array that stores the current temp setting.

The circular array is shared by multiple tasks namely the ISR and the temperature simulator task, the writes to the circular array and increment of index are made atomic for this reason by using `OS_CRITICAL_ENTER` and `OS_CRITICAL_EXIT` commands to disable the ISR when the common memory is been written into. The average temp task also uses the values from the circular array for calculating the average, the interrupts are disabled in that task as well, to ensure its operations are atomic (indivisible).

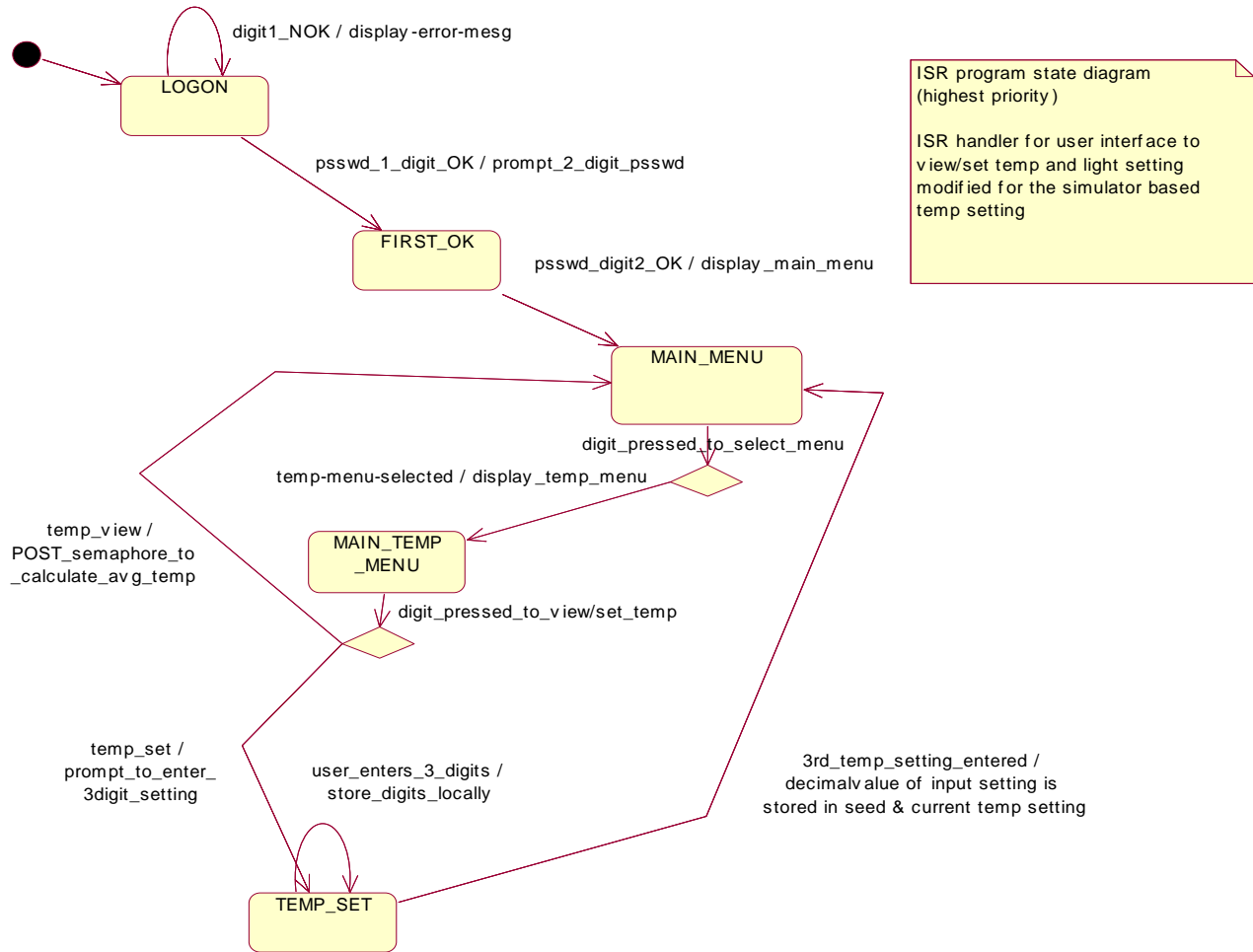
The `OSTimeTickHook` routine in `os_cpu_c.c` was used for scheduling the periodic tasks, and for resetting the received pio-data after 150 ticks – 0.75 seconds, this takes care of enabling multiple digit detection based on this threshold.

The state diagrams and pseudo code for the various modules are given below:

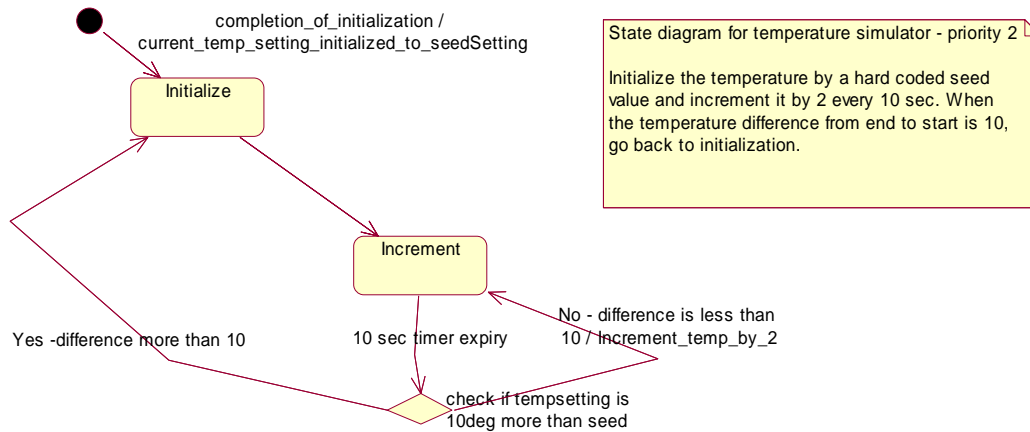
State Diagrams:

ISR Handler –state diagram

Figure 1: State diagram of ISR handler for temp simulator



Temperature simulator – state diagram



Pseudo-Code:

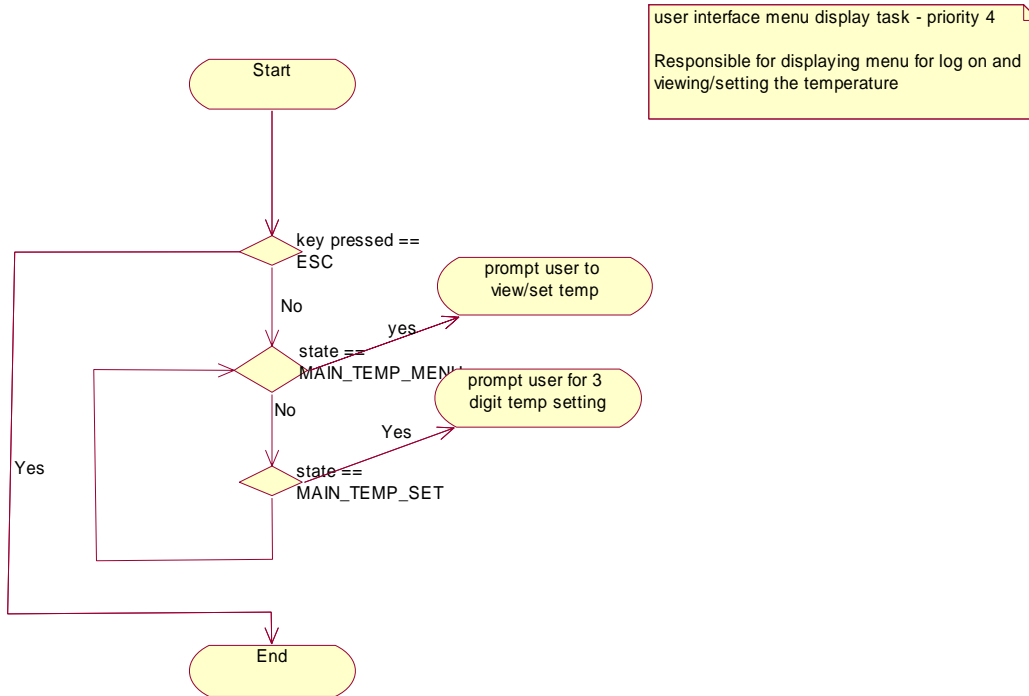
Bubble sort – pseudo code

```

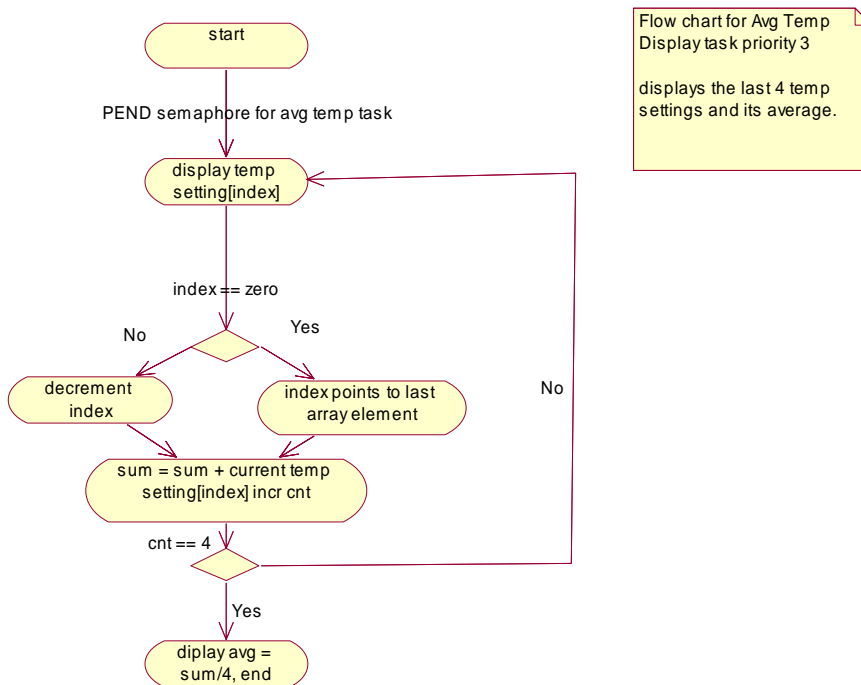
void bubbleSort(int numbers[], int array_size)
while not at end of list
compare adjacent elements
if second is greater than first
switch them
get next two elements
if elements were switched
repeat for entire list
print the first 10 sorted numbers in the array

```

Flow_Chart For User Menu Display



Flow-Chart For Avg Temp Display



Implementation:

//lab7.h –Header file

//defines used by user interface FSM

```
// States the main program has to execute
#define LOGON 51
#define FIRST_OK 52
#define MAIN_MENU 53
#define MAIN_TEMP_MENU 54
#define MAIN_LIGHT_MENU 55
#define MAIN_TEMP_VIEW 56
#define MAIN_TEMP_SET 57
#define MAIN_LIGHT_VIEW 58
#define MAIN_LIGHT_SET 59

#define TEMP_MENU 5
#define LIGHT_MENU 6
#define TEMP_VIEW 7
#define TEMP_SET 8
#define LIGHT_VIEW 3
#define LIGHT_SET 4

#define NO_DIGITS_IN_SETTING 3

// password for nisheet '19'
#define NG_PSSWD_DIG1 1
#define NG_PSSWD_DIG2 9
// password for narayan '27'
#define KN_PSSWD_DIG1 2
#define KN_PSSWD_DIG2 7
```

//defines used in ISR for decoding digit

```
#define PIO_DATA_MASK 0xFF
#define PIO_DATA_ROW_MASK 0xF0 //rows on msn
#define PIO_DATA_COL_MASK 0x0F //columns on lsn

// note row msn col msn make this up
//r4--1,c4--1
#define ONE 17 //00010001
#define TWO 18 //00010010
#define THREE 20 //00010100
#define FOUR 33 //00100001
#define FIVE 34 //00100010
#define SIX 36 //00100100
#define SEVEN 65 //01000001
#define EIGHT 66 //01000010
#define NINE 68 //01000100
#define ZERO_DIGIT 130 //10000010
```

//defines used temperature setting simulator

```

//states of simulator
#define INIT 61
#define INCR 62

// array used to store Temp setting
#define INCR_CNT 6

// Bubble sort array size
#define arr_size 1000

#ifdef     ONT_GLOBALS
#define     ONT_EXT
#else
#define     ONT_EXT  extern
#endif

// to include code to support temp setting simulator

#define SIMULATOR 1

```

//multitask.c file:

```

#include     "./os_cpu.h"
#include     "./os_cfg.h"
#include     "./ucos_ii.h"
#include     "nios.h"
#include     "lab7.h"
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include "pio_lcd16207.h"

//*****
// DATA TYPES
//*****
typedef struct {
    char    TaskName[30];
    INT16U  TaskCtr;
    INT16U  TaskExecTime;
    INT32U  TaskTotExecTime;
} TASK_USER_DATA;

//*****
// VARIABLES
//*****
ONT_EXT  TASK_USER_DATA  TaskUserData[10];

//*****
// FUNCTION PROTOTYPES
//*****
void DispTaskStat(INT8U id);
void bubbleSort  (int numbers[], int array_size);

```

```

void printArray (int nos[]);
void outPutToLCD (int numbers[]);

// allocate memory for tasks' stacks
#define STACKSIZE 2048
OS_STK Stack1[STACKSIZE];
OS_STK Stack2[STACKSIZE];
OS_STK Stack3[STACKSIZE];
OS_STK Stack4[STACKSIZE];
OS_STK Stack5[STACKSIZE];

// global variables used by the tasks of bubblesort and user-interface
OS_EVENT *schSimulator;
OS_EVENT *schMenu;
OS_EVENT *schTime;
OS_EVENT *schAvgTemp;

INT32U decodedDigit;
#ifndef SIMULATOR //enables code for simulation of tempsetting
INT32U state = LOGON;
#else
INT32U state = MAIN_TEMP_MENU;
#endif
INT32U simulatorState = INIT;//used by simulator to inc set temp by 2deg
static int digitCnt = 0; //used by isr to store 3digits of tempsetting
static int incrCnt = 0;

int tempSetting[] = {0,0,0}; //used by isr to store 3digits of tempsetting
#ifdef SIMULATOR //enables code for simulation of tempsetting
int avgTempSetting [INCR_CNT]={0,0,0,0,0,0}; // to store the temp settings of the
simulator
int seedTemp=125; //hardcoded initial tempsetting
#endif
int lightSetting[] = {0,0,0}; //used by isr to store 3digits of lightsetting

int oNum[1000]; //array used for bubblesorting

int pio_data = 0; //row and col values of keypad_pio_isr
int rcvd_pio = 0; //prevents multiplr isr's(due to one-hot-cntr) being
processed

```

//ten second –system time task –priority 1

```

void tenSecTask (void *i)
{
    INT32U Time;
    INT8U Reply;

    char str[]={'\0'};
    CPUInit(); // highest priority task - initialize and enable timer
    for (;;) // infinite loop
    {
        OSSemPend(schTime, 0, &Reply);
        // print time (clock ticks)
        Time = OSTimeGet();
        printf( "Time Task_system_time: %d\n", Time );
        str[0]='\0';
    }
}

```

```

    sprintf(str,"The System Time:%d",Time); nr_pio_lcdwritescree(str);
    nr_delay(1000);    //displays time for a 1 second
}
}

```

//temperature-simulator task –priority 2

```

void tempSetSimulator(void *i)
{
    INT32U  Time;
    INT8U   Reply;
    int j,t;
    int tempVal=0;
    for (;;) // infinite loop
    {

        OSSEmpend(schSimulator, 0, &Reply);
        OS_ENTER_CRITICAL()
        tempVal =0;

        Time = OSTimeGet();
        printf( "Simulator Task: %d\n", Time );
        if(simulatorState == INIT)
        {

            //avgTempSetting[incrCnt++] = (rand()%1000);

            avgTempSetting[incrCnt++] = seedTemp;
            printf("incrcnt:%d tempval:%d tickCnt:%d\n",incrCnt,seedTemp,Time);
            simulatorState = INCR;
            if(incrCnt == INCR_CNT)
                incrCnt = 0;                //overflow check of array-index
        }
        else if(simulatorState == INCR)
        {
            if(incrCnt > 0)                //underflow check of array-index
                tempVal = avgTempSetting[incrCnt-1];
            else
                tempVal = avgTempSetting[INCR_CNT-1];

            tempVal += 2;                // increment temp setting by 2
            printf("incrcnt:%d tempval:%d tickCnt:%d\n",incrCnt,tempVal,Time);
            avgTempSetting[incrCnt++] = tempVal;

            if(incrCnt == INCR_CNT)
                incrCnt = 0;

            if(tempVal == (seedTemp+10))
                simulatorState = INIT;
        }
        else
            printf("Error in simulator state\n");
        OS_EXIT_CRITICAL();
    } // for
}

```

//temp setting view task –avg of latest four settings –priority 3

```

void avgTempDisplay (void *i)
{
    INT32U  Time;
    INT8U   Reply;
    int j,n;
    int sum=0;
    char str[]={'\0'};
    for (;;) // infinite loop
    {
        OSSemPend(schAvgTemp, 0, &Reply);
        OS_ENTER_CRITICAL();

        Time = OSTimeGet();
        sum = 0;
        j = incrCnt;
        for(n= 1; n< 5;n++)
            if(!(j))
            {
                j= INCR_CNT-1;
                printf("the array avgTemp:%d\n",avgTempSetting[j]) ;
                sum += avgTempSetting[j] ;
            }
            else
            {
                j= --j;
                printf("the array avgTemp:%d\n",avgTempSetting[j]) ;
                sum += avgTempSetting[j] ;
            }

        printf( "Tick:%d Avg Temp Task Avg_temp is: %d\n",Time, (sum/4) );
        str[0]='\0';
        sprintf(str,"The avg Temp      : %d", (sum/4) );
        nr_pio_lcdwritescreen(str);
        OS_EXIT_CRITICAL();
    }
}

```

//user interface -menu display task –priority 4

```

void backgroundMenu(void *i)
{
    INT32U  Time;
    INT8U   Reply;
    for (;;) // infinite loop
    {
        OSSemPend(schMenu, 0, &Reply);
        //Time = OSTimeGet();
        //printf( "user interface Task: %d\n", Time );

        switch(state)
        {
            case LOGON:
                // nr_delay(1000);

```

```

    nr_pio_lcdwritescreen("Please Enter    the Password");

    break;
case FIRST_OK:
    //nr_delay(1000);
    nr_pio_lcdwritescreen("Please Enter    second digit");

    break;
case MAIN_MENU:
    //nr_delay(3000);
    nr_pio_lcdwritescreen("Press 5 for Temp Press 6for Light");

    break;
case MAIN_TEMP_MENU:
    //nr_delay(1000);
    nr_pio_lcdwritescreen("Press 7for  view, Press 8for Set");
    break;
case MAIN_TEMP_SET:
    //nr_delay(1000);
    nr_pio_lcdwritescreen("Enter 3digit    Temp Setting");
    break;
case MAIN_LIGHT_MENU:
    //nr_delay(1000);
    nr_pio_lcdwritescreen("Press 3for  view, Press 4for Set");
    break;
case MAIN_LIGHT_SET:
    //nr_delay(1000);
    nr_pio_lcdwritescreen("Enter 3digit    Light Setting");
    break;
default:
    printf("error in main state\n");
    break;
}
}
}

```

// background -bubble sorting task –priority 5

```

void bubbleSortTask(void *w)
{
    INT32U  Time;
    INT8U   Reply;

    //for (;;) // infinite loop
    // {
        int seed;// seed to initialize the random no. generator with.
        int r; /* random value in range */
        int M; /* user supplied upper boundary */
        int x;
        int arrNum[1000];
        int count;
    /* random int in range [0,M) if M is an integer then range = [0,M-1] */
        int y;
        int choice=0;
        int i,as;

```

```

seed = rand()%100; /* choose a random seed value */
srand(seed); /*initialize random number generator*/

printf("init random array\n");
for(count=0; count<1000; ++count)
    { //initializes random array of nos.to be sorted
    x = (rand()%1000); arrNum[count]=x;
    // printf("%d \n",x);
    // printf("%d \n",arrNum[count]);
    }
as = arr_size;
nr_pio_lcdinit(na_lcd_pio); // initialize the lcd to display

for(i=0;i<arr_size;i++)
    oNum[i] = arrNum[i]; // save random array in oNum

//printf("Unsorted Array\n");
//printArray(oNum);
//printf("Sorting using Bubble Sort\n");
bubbleSort(arrNum,as);
OSTaskDel(OS_PRIO_SELF); // delete task after completion
// }//for
}

```

// bubbleSort and its associated routines

```

void bubbleSort(int numbers[], int array_size)
{
    int i=0;
    int j, temp;

    for (i = (arr_size - 1); i >= 0; i--)
    {
        for (j = 1; j <= i; j++)
        {
            if (numbers[j-1] > numbers[j])
            {
                temp = numbers[j-1];
                numbers[j-1] = numbers[j];
                numbers[j] = temp;

                // nr_delay(1400);
            }
        }
    }
    //printArray(numbers);
    outPutToLCD(numbers);
    printf("sorted first 10
nos:%d_%d_%d_%d_%d_%d_%d_%d_%d_%d\n", numbers[0], numbers[1], numbers[2], numbers[3], n
umbers[4], numbers[5], numbers[6], numbers[7], numbers[8], numbers[9]);
}

void outPutToLCD (int n[])
{ // form the 32 character string

```

```

    char str[128];

    sprintf(str,"%d_%d_%d_%d_%d_%d_%d_%d_%d",n[0],n[1],n[2],n[3],n[4],n[5],n[6],n[7],n[8],n[9]);
    nr_pio_lcdwritescree(str);
    //nr_delay(1000);          // displays for a second
    // sprintf(str,"%d %d %d %d %d",n[5],n[6],n[7],n[8],n[9]);
    //nr_pio_lcdwritescree(str);
}

void printArray(int nos[])
{
    int i;
    for(i =0;i<arr_size;i++)
        printf("%d ",nos[i]);

    printf("\n");
}

```

//ISR handler –decodes digit & cntrl’s state transitions of user interface

```

void keypadPIO_ISR(int context)
{
    np_pio *pio = (np_pio *)context;

    int decodedDigit;
    char strDigit[]={'\0','\0','\0','\0'};
    int i;
    int *ptr;
    nr_delay(5); // 10 msec delay to take care of transients and debounce

    pio_data = pio->np_piodata;
    pio_data = pio_data & PIO_DATA_MASK;

    // nr_pio_lcdwritescree("enter ISR");
    // printf("the new piodata:%d\n", pio_data);

    // prevent isr processing repeatedly when key remains pressed continuously
    if(rcvd_pio != pio_data && (pio_data & PIO_DATA_COL_MASK))
    {
        rcvd_pio = pio_data;

        //printf("the state is:%d\n",state);
        //printf("rec pio:%d \n",rcvd_pio);

        decodedDigit= decodeDigit();

        switch(state)
        {
        case LOGON:
            if((decodedDigit == NG_PSSWD_DIG1)||((decodedDigit == KN_PSSWD_DIG1))
            {
                state = FIRST_OK;
                printf("Please Enter    second digit\n");
            }
        }
    }
}

```

```

else // display psswd rejected and prompt again!
{
nr_pio_lcdwritescreen("Incorrect password try again!");
printf("Incorrect password try again!\n");
state=LOGON;
}
break; // no processing for first digit
case FIRST_OK:
if((decodedDigit == NG_PSSWD_DIG2)|| (decodedDigit == KN_PSSWD_DIG2))
{
state = MAIN_MENU;
printf("Press 5 for Temp Press 6for Light\n");
}
else // display psswd rejected and prompt again!
{
nr_pio_lcdwritescreen("Incorrect password try again!");
printf("Incorrect password try again!\n");
state=LOGON;
}
break;
case MAIN_MENU:
if(decodedDigit == TEMP_MENU)
{
state = MAIN_TEMP_MENU;
printf("Enter 7-to view,8-to change Temp\n");
}
else if (decodedDigit == LIGHT_MENU)
{
state = MAIN_LIGHT_MENU;
printf("Enter 3-to view,4-to change light\n");
}
else
state=MAIN_MENU; // invalid entry takes you back to the main menu
break;
case MAIN_TEMP_MENU:
if(decodedDigit == TEMP_VIEW)
{
#ifdef SIMULATOR //ENABLES CODE FOR SIMULATION OF TEMPSETTING
strDigit[0]='\0'; ptr = &tempSetting[0];
printf("Temp Setting is %d%d%d\n",*ptr++,*ptr++,*ptr);
ptr = &tempSetting[0]; // ptr to first element in the array
sprintf(strDigit,"Temp Setting is %d%d%d",*ptr++,*ptr++,*ptr);
nr_pio_lcdwritescreen(strDigit);

state = MAIN_MENU;
printf("Press 5 for Temp Press 6for Light\n");
#else

OSSemPost(schAvgTemp); //enable average calculation
printf("Press 7for view, Press 8for Set\n");
#endif
}
else if(decodedDigit == TEMP_SET)
{
state = MAIN_TEMP_SET;
printf("Enter 3 digit Temp Setting\n");
}
#endif
}

```

```

        /* NOTE:
        *IDENTICAL INPUTS,condition has been taken care by using
        *a timerTick to reset rcvd_pio data*/
#endif
    }
    break;
case MAIN_TEMP_SET:
    tempSetting[digitCnt++] = decodedDigit;
    if(digitCnt == NO_DIGITS_IN_SETTING)
    {
        digitCnt=0;
#ifdef SIMULATOR //ENABLES CODE FOR SIMULATION OF TEMPSETTING
        ptr = &tempSetting[0];
        printf("Temp Setting is %d%d%d\n",*ptr++,*ptr++,*ptr);
        strDigit[0] = '\0'; ptr = &tempSetting[0];
        sprintf(strDigit,"Temp Setting is %d%d%d",*ptr++,*ptr++,*ptr);
        nr_pio_lcdwritescree(strDigit);

        state = MAIN_MENU;
        printf("Press 5 for Temp Press 6for Light\n");
#else
        seedTemp = (tempSetting[0]*100 + tempSetting[1]*10 +
tempSetting[2]);
        avgTempSetting[incrCnt]= seedTemp;
        incrCnt = (++incrCnt)%INCR_CNT;

        state = MAIN_TEMP_MENU;
        printf("Press 7for view, Press 8for Set\n");
#endif
    }
    break;

case MAIN_LIGHT_MENU:
    if(decodedDigit == LIGHT_VIEW)
    {
        strDigit[0] = '\0'; ptr = &lightSetting[0];
        printf("Light Setting is %d%d%d",*ptr++,*ptr++,*ptr);
        ptr = &lightSetting[0];
        sprintf(strDigit,"Light Setting is %d%d%d",*ptr++,*ptr++,*ptr);
        nr_pio_lcdwritescree(strDigit);

        state = MAIN_MENU;
        printf("Press 5 for Temp Press 6for Light\n");
    }
    else if(decodedDigit == LIGHT_SET)
        state = MAIN_LIGHT_SET;
    printf("Enter 3digit Light Setting\n");
#ifdef 0
        /* NOTE:
        *IDENTICAL INPUTS,condition has been taken care by using
        *a timerTick to reset rcvd_pio data*/
#endif
    break;
case MAIN_LIGHT_SET:
    lightSetting[digitCnt++] = decodedDigit;
    if(digitCnt == NO_DIGITS_IN_SETTING)
    {

```

```

        digitCnt=0; ptr = &lightSetting[0];
        printf("Light Setting is %d%d%d\n",*ptr++,*ptr++,*ptr);
        strDigit[0] = '\0'; ptr = &lightSetting[0];
        sprintf(strDigit,"Light Setting is %d%d%d",*ptr++,*ptr++,*ptr);
        nr_pio_lcdwritescreen(strDigit);

        state = MAIN_MENU;
        printf("Press 5 for Temp Press 6for Light\n");
    }
    break;
} //switch

} //if
pio->np_pioedgecapture = 0; // clear the irq condition
}

```

//decodeDigit routine called by ISR handler

```

int decodeDigit()
{int dDigit;

    switch(pio_data)
    {
        case ONE:
            dDigit = 1;
            break;
        case TWO:
            dDigit = 2;
            break;
        case THREE:
            dDigit = 3;
            break;
        case FOUR:
            dDigit = 4;
            break;
        case FIVE:
            dDigit = 5;
            break;
        case SIX:
            dDigit = 6;
            break;
        case SEVEN:
            dDigit = 7;
            break;
        case EIGHT:
            dDigit = 8;
            break;
        case NINE:
            dDigit = 9;
            break;
        case ZERO_DIGIT:
            dDigit = 0;
            break;
        default:
            // only 0-9 decoded other keys given hex e value
            dDigit = 0xe;
    }
}

```

```

        break;
    }
    printf("the digit is: %d\n", dDigit);
    return dDigit;
}

```

//main module – task & semaphore creation, ISR,LCD & OS initialization

```

// Main function.
int main(int argc, char **argv)
{
    char Id1 = '1';
    char Id2 = '2';
    char Id3 = '3';
    char Id4 = '4';
    char Id5 = '5';
    np_pio *pio_pointer = na_keypad_pio;
    // For use with debug
    #if NIOS_GDB
        nios_gdb_install(1);
        nios_gdb_breakpoint();
    #endif
    // init the LCD
    nr_pio_lcdinit(na_lcd_pio);
    //ISR init, setting up our keypad_pio isr

    nr_installuserisr(na_keypad_pio_irq,keypadPIO_ISR, (int)pio_pointer);

    //Mask 0-7bits for data
    pio_pointer->np_piodirection = 0; // all 8 lines are input
    pio_pointer->np_pioedgecapture = 0; // clears any previous isr's
    // use colum bits for interrupt generation
    pio_pointer->np_piointerruptmask = PIO_DATA_COL_MASK;

    // needed by uC/OS
    OSInit();
    schSimulator = OSSemCreate(1);
    schMenu      = OSSemCreate(1);
    schTime      = OSSemCreate(1);
    schAvgTemp   = OSSemCreate(0);

    // create the tasks in uC/OS and assign decreasing priority to them
    OSTaskCreate(tenSecTask,      (void *)&Id1, (void *)&Stack1[STACKSIZE-1], 1);
    OSTaskCreate(tempSetSimulator, (void *)&Id2, (void *)&Stack2[STACKSIZE-1], 2);
    OSTaskCreate(avgTempDisplay,  (void *)&Id3, (void *)&Stack3[STACKSIZE-1], 3);
    OSTaskCreate(backgroundMenu,  (void *)&Id4, (void *)&Stack4[STACKSIZE-1], 4);
    OSTaskCreate(bubbleSortTask,  (void *)&Id5, (void *)&Stack5[STACKSIZE-1], 5);
    // start the os
    OSStart();
}

```

//OSTimeTickHook module in os_cpu_c.c –nr_timer1 based scheduling

```

void OSTimeTickHook (void)
{
static int tempSimTick = 2000;    // run every 10 secs
static int menuTick    = 400;    // run every 2 secs
static int timeTick    = 2000;   // run every 10 secs
static int pioTick     = 150;    // hold rcv_pio for 0.75 secs

ONT_EXT OS_EVENT *schSimulator;
ONT_EXT OS_EVENT *schMenu;
ONT_EXT OS_EVENT *schTime;
ONT_EXT OS_EVENT *schAvgTemp;   // used for signalling avgTempDisplay task

extern int rcvd_pio;
if(rcvd_pio)                // to reset rcv_pio_data only after 150 ticks
    pioTick--;
if(!pioTick)
{
    rcvd_pio = 0;          // reset rcv_pio_data to accept new inputs
    pioTick = 150;
}

tempSimTick--;
if(!tempSimTick)
{
    OSSemPost(schSimulator);
    tempSimTick = 2000;
}

timeTick--;
if(!timeTick)
{
    OSSemPost(schTime);
    timeTick = 2000;
}

menuTick--;
if(!menuTick)
{
    OSSemPost(schMenu);
    menuTick = 400;
}
}

```

Validation and Testing:

The validation and testing of the entire lab 7 was carried out in incremental steps, the multiple tasks were integrated one at a time and checked for functionality individually.

The following white box testing was carried out to check for stability and robustness of all paths of the software:

- Temp simulator functionality – increment up to 10degrees difference and its wrap around for the seed value was checked
- Circular operation of the indices (checks for overflow and underflow) done in the average temp display task
- Inter task signaling checked between the ISR and average temp display task.
- Bubble sort routine was checked for generation and sorting of 10000 and 1000 array of numbers.
- Integration testing and

Results:

The user interface code written for lab assignment #6 and the bubble sort written as part of lab assignment #5 were combined into a single piece of code. A new routine for simulating varying temperature readout was written. The program started out by using a hard coded seed value for the temperature and then incremented it by 2 degrees every 10 seconds. When the temperature deviated by a margin of 10 degrees, the temperature rolled back to the seed value. During anytime, the user was able to set a new temperature seed value. If the user chose to view the temperature value, he/she was shown the average of the last four temperature values. After the bubble sort routine completed sorting 100,000 values, the first 10 values were printed on the LCD screen. The following output was seen on the Nios-SDK shell window.

SPACE INTENTIONALLY LEFT BLANK

<pre> nios-run: Terminal mode (Control-C exits) ----- HL: 224 HL: 224 HL: 224 HL: 224 HL: 224 HL: 224 HL: 224 Time Task_system_time: 0 Simulator Task: 253 incrCnt:1 tempVal:125 tickCnt:253 init random array sorted first 10 nos:0_1_1_3_3_3_6_7_8_8 Time Task_system_time: 2000 Simulator Task: 2253 incrCnt:1 tempVal:127 tickCnt:2253 the digit is: 7 the array avgTemp:127 the array avgTemp:125 the array avgTemp:0 the array avgTemp:0 Tick:2304 Avg Temp Task Avg_temp is: 63 Press 7for view, Press 8for Set Time Task_system_time: 4000 Simulator Task: 4253 incrCnt:2 tempVal:129 tickCnt:4253 the digit is: 7 the array avgTemp:129 the array avgTemp:127 the array avgTemp:125 the array avgTemp:0 Tick:4453 Avg Temp Task Avg_temp is: 95 Press 7for view, Press 8for Set Time Task_system_time: 6000 Simulator Task: 6253 incrCnt:3 tempVal:131 tickCnt:6253 the digit is: 8 Enter 3 digit Temp Setting the digit is: 4 the digit is: 5 the digit is: 5 Press 7for view, Press 8for Set the digit is: 9 Time Task_system_time: 8000 Simulator Task: 8253 incrCnt:5 tempVal:457 tickCnt:8253 the digit is: 7 the array avgTemp:457 the array avgTemp:455 the array avgTemp:131 the array avgTemp:129 Tick:9310 Avg Temp Task Avg_temp is: 293 </pre>	<pre> Press 7for view, Press 8for Set Time Task_system_time: 10000 Simulator Task: 10253 incrCnt:0 tempVal:459 tickCnt:10253 the digit is: 8 Enter 3 digit Temp Setting Time Task_system_time: 12000 the digit is: 5 the digit is: 6 Simulator Task: 12301 incrCnt:1 tempVal:461 tickCnt:12301 the digit is: 3 Press 7for view, Press 8for Set Time Task_system_time: 14000 the digit is: 7 Press 7for view, Press 8for Set Simulator Task: 14270 incrCnt:3 tempVal:565 tickCnt:14270 the array avgTemp:565 the array avgTemp:563 the array avgTemp:461 the array avgTemp:459 Tick:14271 Avg Temp Task Avg_temp is: 512 Time Task_system_time: 16000 Simulator Task: 16253 incrCnt:4 tempVal:567 tickCnt:16253 Time Task_system_time: 18000 Simulator Task: 18253 incrCnt:5 tempVal:569 tickCnt:18253 Time Task_system_time: 20000 Simulator Task: 20253 incrCnt:0 tempVal:571 tickCnt:20253 Time Task_system_time: 22000 Simulator Task: 22253 incrCnt:1 tempVal:573 tickCnt:22253 Time Task_system_time: 24000 Simulator Task: 24253 incrCnt:3 tempVal:563 tickCnt:24253 the digit is: 8 Enter 3 digit Temp Setting the digit is: 6 the digit is: 9 the digit is: 8 Press 7for view, Press 8for Set Time Task_system_time: 26000 Simulator Task: 26253 incrCnt:4 tempVal:700 tickCnt:26253 the digit is: 7 the array avgTemp:700 the array avgTemp:698 the array avgTemp:563 the array avgTemp:573 Tick:26502 Avg Temp Task Avg_temp is: 633 Press 7for view, Press 8for Set Time Task_system_time: 28000 </pre>
---	--

Conclusion:

In this lab assignment, some very important concepts and fundamentals of MicroC/OS-II Real Time Operating System were learned. Specifically, the importance of carefully choosing task priorities, semaphore synchronization, task suspension and resume, and semaphore posting and pending were learned. Also, the issue of pressing same key in a row that was not implemented in lab 6 was addressed in this lab.