Configuration of Nios™ Soft Core & Peripherals
and
Executing Software on a Nios™ Embedded Processor

LAB #4-5 EE 2160
October 11, 2004

BY:
Narayanan Krishnamurthy

TABLE OF CONTENTS

Purpose:

The purpose of lab 4 was to implement a 32 bit Nios Soft Core and peripherals using Quartus II. The reference design was used to expedite the process of adding and assigning pins. In lab 5, C code was created, compiled, and executed on a Nios embedded processor running on an Altera Excalibur board. Nios SDK Unix shell was used to execute and download the code on the FPGA board.

Procedure:

The Nios embedded processor was created using the instructions on the lab handout and the Nios Tutorial provided as part of the lab. Following steps outlay the processor creation process:

1. A top-level Block Design file (.bdf) called **nios_system_module** was created.

2. The MegaWizard Plug-In Manager was then used to launch the Altera SOPC Builder 2.8. The SOPC Builder allows for the creation of the Nios CPU and peripherals.

3. The following modules were then added to the SOPC Builder:
   - Nios 32-Bit CPU
   - Boot Monitor ROM
   - Communications UART
   - Debugging UART
   - Timer
   - Button PIO
   - LCD PIO
   - LED PIO
   - Seven Segment PIO
   - External RAM Bus (Avalon Tri-State Bridge)
   - External RAM Interface
   - External Flash Interface

4. The base address and IRQs were then verified using Figure 27 of the Nios Tutorial. Please refer to Figure 1 in the design section for the various parameters in Nios Processor and peripherals.

5. The design was then synthesized using the Generate command in the SPOC Builder.

6. After the generation of the design, the symbol for the Nios system module was added to the BDF.

The input and output pin connections to the CPU was made in the .BDF file and compiled. The .CSF file was edited to create pin assignments the reference design and the reference .CSF file were used for this purpose. For each I/O pin appearing in the schematic, the corresponding pin assignment statement were copied from the reference design .CSF file into he same place in the .CSF file for the lab project.

Programmer command was then used to download the Nios Processor design into the APEX20KE FPGA in the Excalibur target board.

Nios SDK shell was then used to run the nios-build and nios-run utilities on the Nios development board. As a test for a correct implementation of the Nios processor, the hello_nios file was downloaded and tested on the FPGA board. Following two commands were used to first build and then run output of the file:

```
nb hello_nios.c ↵
nr hello_nios.srec ↵
```

The program generated the message “Hello, from Nios!” and caused the 2-digit 7-segment LED to count down from 99 to 0.

As part of the assignment, a C program was written to implement bubble sort and selection sort. The program took a set of 10 unsorted integer values as inputs and sorted them by employing the Selection Sort and the Bubble Sort technique. The output of the programs, showing the values being sorted step-by-step was displayed to both the monitor and the LCD display. Please refer to the design section for the source code and see figure 3 in the results section for the program output.

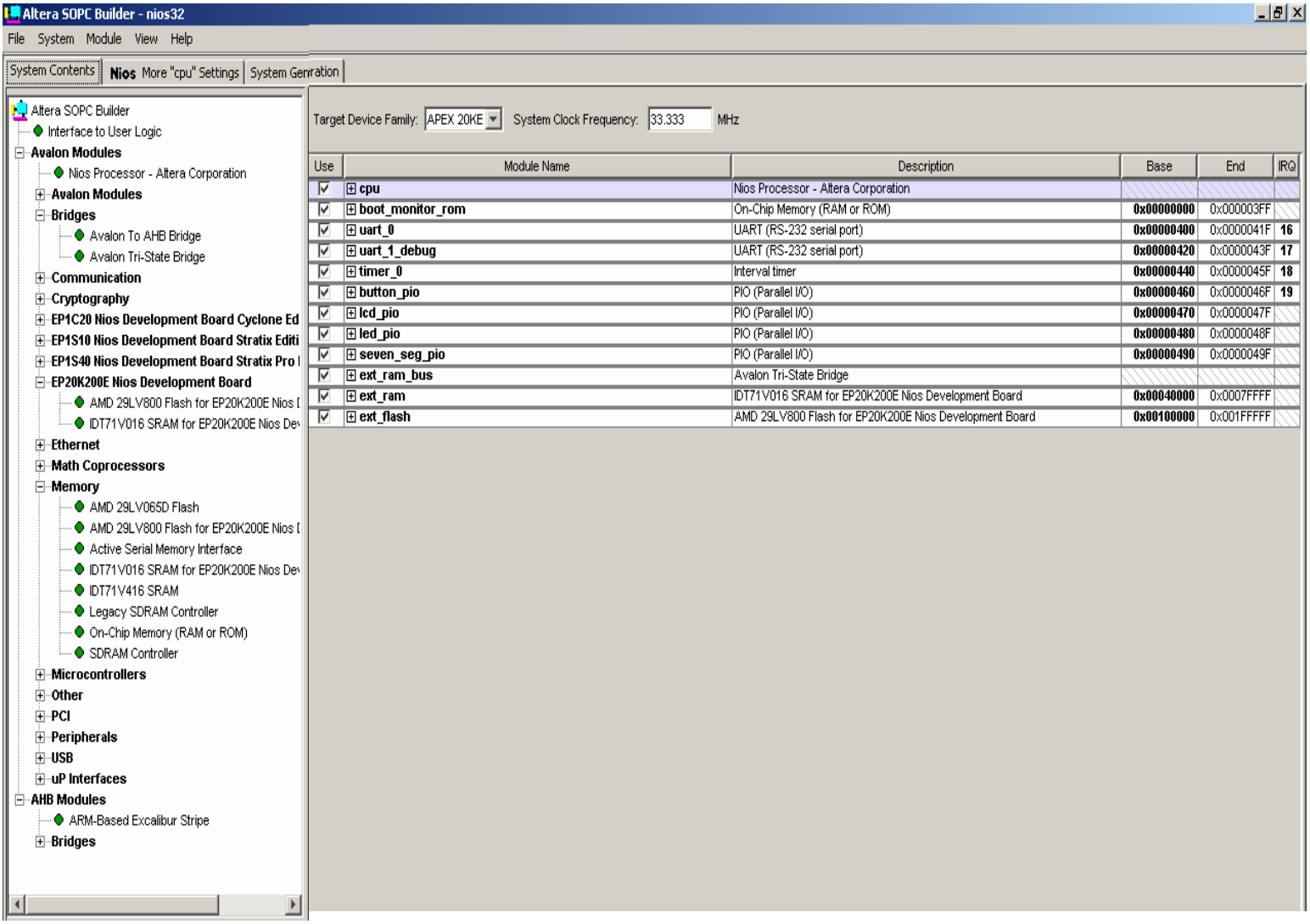
Design:

Nios softcore processor and its peripherals: The Altera-Excalibur's tutorial was used for creating the 32-bit Nios 33.33 Mhz processor. The related peripherals including ROM, RAM, and the other peripherals listed below were also created as per the tutorial.

■ Boot Monitor ROM ■ UART_0 ■ UART_1_debug ■ Timer_0 ■ Button PIO	■ LCD PIO ■ LED PIO ■ Seven Segment PIO ■ External RAM Bus (Avalon Tri-State Bridge) ■ External RAM Interface ■ External Flash Interface

Note: Check for the IRQ interrupt nos(16-urart_0,17-uart_1_debug,18-timer_0 and 19-button_pio, the priority has to be maintained in this order for the processor to function properly.

Figure 1: Nios Processor and Peripherals



Pin assignments: The 32-bit Nios reference design was used to simplify the process of pin assignments, the following pin details of the reference design was copied into the CHIP module of our nios_system_module .CSF file.

Note : Take care in mapping the IO pins to the created processor symbol(check the reference-32bit-Nios design) before generating your .CSF file. Confirm the following options in the CHIP module, of the nios_system_module.csf file, `DEVICE = "EP20K200EFC484-2X";`
`RESERVE_ALL_UNUSED_PINS = "AS INPUT TRI-STATED";`

```
CLK_DRV/clk_to_apex : LOCATION = Pin_L6;
DB9_CONNECTOR/rxd : LOCATION = Pin_W8;
DB9_CONNECTOR/txd : LOCATION = Pin_D15;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[0]" : LOCATION = Pin_Y9;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[10]" : LOCATION = Pin_U12;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[11]" : LOCATION = Pin_Y10;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[1]" : LOCATION = Pin_T9;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[2]" : LOCATION = Pin_Y8;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[3]" : LOCATION = Pin_W9;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[4]" : LOCATION = Pin_V9;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[5]" : LOCATION = Pin_U9;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[6]" : LOCATION = Pin_T10;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[7]" : LOCATION = Pin_U10;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[8]" : LOCATION = Pin_V10;
"DIP_SWITCH/all-SW7/out-SW6/out-SW5/out-SW4/out[9]" : LOCATION = Pin_P11;
"DISPLAY/tens_digit-DISPLAY/ones_digit[0]" : LOCATION = Pin_W17;
"DISPLAY/tens_digit-DISPLAY/ones_digit[10]" : LOCATION = Pin_Y17;
"DISPLAY/tens_digit-DISPLAY/ones_digit[11]" : LOCATION = Pin_V8;
"DISPLAY/tens_digit-DISPLAY/ones_digit[12]" : LOCATION = Pin_Y7;
"DISPLAY/tens_digit-DISPLAY/ones_digit[13]" : LOCATION = Pin_U11;
"DISPLAY/tens_digit-DISPLAY/ones_digit[14]" : LOCATION = Pin_R11;
"DISPLAY/tens_digit-DISPLAY/ones_digit[15]" : LOCATION = Pin_D18;
"DISPLAY/tens_digit-DISPLAY/ones_digit[1]" : LOCATION = Pin_U18;
"DISPLAY/tens_digit-DISPLAY/ones_digit[2]" : LOCATION = Pin_Y18;
"DISPLAY/tens_digit-DISPLAY/ones_digit[3]" : LOCATION = Pin_W18;
"DISPLAY/tens_digit-DISPLAY/ones_digit[4]" : LOCATION = Pin_U8;
"DISPLAY/tens_digit-DISPLAY/ones_digit[5]" : LOCATION = Pin_T11;
"DISPLAY/tens_digit-DISPLAY/ones_digit[6]" : LOCATION = Pin_R10;
"DISPLAY/tens_digit-DISPLAY/ones_digit[7]" : LOCATION = Pin_C18;
"DISPLAY/tens_digit-DISPLAY/ones_digit[8]" : LOCATION = Pin_V17;
"DISPLAY/tens_digit-DISPLAY/ones_digit[9]" : LOCATION = Pin_V18;
FLASH/a16 : LOCATION = Pin_F3;
FLASH/ce_n : LOCATION = Pin_E1;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[0]" : LOCATION = Pin_G17;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[10]" : LOCATION = Pin_A3;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[11]" : LOCATION = Pin_A4;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[12]" : LOCATION = Pin_C3;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[13]" : LOCATION = Pin_C1;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[14]" : LOCATION = Pin_D3;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[15]" : LOCATION = Pin_D2;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[16]" : LOCATION = Pin_C2;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[17]" : LOCATION = Pin_D1;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[18]" : LOCATION = Pin_B3;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[19]" : LOCATION = Pin_E3;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[1]" : LOCATION = Pin_A8;
```

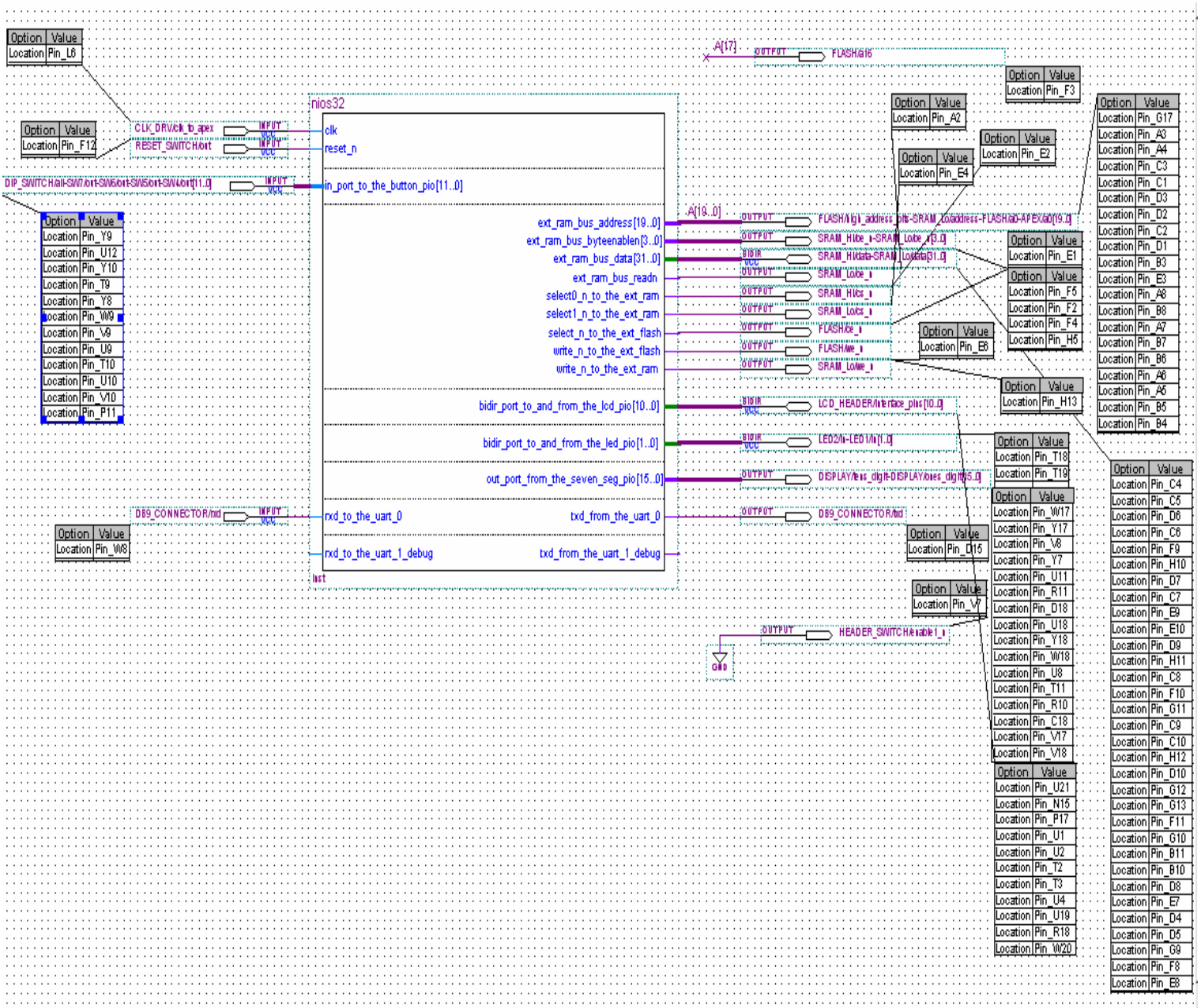
```
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[2]" : LOCATION = Pin_B8;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[3]" : LOCATION = Pin_A7;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[4]" : LOCATION = Pin_B7;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[5]" : LOCATION = Pin_B6;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[6]" : LOCATION = Pin_A6;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[7]" : LOCATION = Pin_A5;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[8]" : LOCATION = Pin_B5;
"FLASH/high_address_bits-SRAM_Lo/address-FLASH/a0-APEX/a0[9]" : LOCATION = Pin_B4;
FLASH/we_n : LOCATION = Pin_H13;
HEADER_SWITCH/enable1_n : LOCATION = Pin_V7;
LCD_HEADER/interface_pins[0] : LOCATION = Pin_U21;
LCD_HEADER/interface_pins[10] : LOCATION = Pin_N15;
LCD_HEADER/interface_pins[1] : LOCATION = Pin_P17;
LCD_HEADER/interface_pins[2] : LOCATION = Pin_U1;
LCD_HEADER/interface_pins[3] : LOCATION = Pin_U2;
LCD_HEADER/interface_pins[4] : LOCATION = Pin_T2;
LCD_HEADER/interface_pins[5] : LOCATION = Pin_T3;
LCD_HEADER/interface_pins[6] : LOCATION = Pin_U4;
LCD_HEADER/interface_pins[7] : LOCATION = Pin_U19;
LCD_HEADER/interface_pins[8] : LOCATION = Pin_R18;
LCD_HEADER/interface_pins[9] : LOCATION = Pin_W20;
"LED2/in-LED1/in[0]" : LOCATION = Pin_T18;
"LED2/in-LED1/in[1]" : LOCATION = Pin_T19;
RESET_SWITCH/out : LOCATION = Pin_F12;
"SRAM_Hi/be_n-SRAM_Lo/be_n[0]" : LOCATION = Pin_F5;
"SRAM_Hi/be_n-SRAM_Lo/be_n[1]" : LOCATION = Pin_F2;
"SRAM_Hi/be_n-SRAM_Lo/be_n[2]" : LOCATION = Pin_F4;
"SRAM_Hi/be_n-SRAM_Lo/be_n[3]" : LOCATION = Pin_H5;
SRAM_Hi/cs_n : LOCATION = Pin_E2;
"SRAM_Hi/data-SRAM_Lo/data[0]" : LOCATION = Pin_C4;
"SRAM_Hi/data-SRAM_Lo/data[10]" : LOCATION = Pin_C5;
"SRAM_Hi/data-SRAM_Lo/data[11]" : LOCATION = Pin_D6;
"SRAM_Hi/data-SRAM_Lo/data[12]" : LOCATION = Pin_C6;
"SRAM_Hi/data-SRAM_Lo/data[13]" : LOCATION = Pin_F9;
"SRAM_Hi/data-SRAM_Lo/data[14]" : LOCATION = Pin_H10;
"SRAM_Hi/data-SRAM_Lo/data[15]" : LOCATION = Pin_D7;
"SRAM_Hi/data-SRAM_Lo/data[16]" : LOCATION = Pin_C7;
"SRAM_Hi/data-SRAM_Lo/data[17]" : LOCATION = Pin_E9;
"SRAM_Hi/data-SRAM_Lo/data[18]" : LOCATION = Pin_E10;
"SRAM_Hi/data-SRAM_Lo/data[19]" : LOCATION = Pin_D9;
"SRAM_Hi/data-SRAM_Lo/data[1]" : LOCATION = Pin_H11;
"SRAM_Hi/data-SRAM_Lo/data[20]" : LOCATION = Pin_C8;
"SRAM_Hi/data-SRAM_Lo/data[21]" : LOCATION = Pin_F10;
"SRAM_Hi/data-SRAM_Lo/data[22]" : LOCATION = Pin_G11;
"SRAM_Hi/data-SRAM_Lo/data[23]" : LOCATION = Pin_C9;
"SRAM_Hi/data-SRAM_Lo/data[24]" : LOCATION = Pin_C10;
"SRAM_Hi/data-SRAM_Lo/data[25]" : LOCATION = Pin_H12;
"SRAM_Hi/data-SRAM_Lo/data[26]" : LOCATION = Pin_D10;
"SRAM_Hi/data-SRAM_Lo/data[27]" : LOCATION = Pin_G12;
"SRAM_Hi/data-SRAM_Lo/data[28]" : LOCATION = Pin_G13;
"SRAM_Hi/data-SRAM_Lo/data[29]" : LOCATION = Pin_F11;
"SRAM_Hi/data-SRAM_Lo/data[2]" : LOCATION = Pin_G10;
"SRAM_Hi/data-SRAM_Lo/data[30]" : LOCATION = Pin_B11;
"SRAM_Hi/data-SRAM_Lo/data[31]" : LOCATION = Pin_B10;
"SRAM_Hi/data-SRAM_Lo/data[3]" : LOCATION = Pin_D8;
"SRAM_Hi/data-SRAM_Lo/data[4]" : LOCATION = Pin_E7;
"SRAM_Hi/data-SRAM_Lo/data[5]" : LOCATION = Pin_D4;
```

```
"SRAM_Hi/data-SRAM_Lo/data[6]" : LOCATION = Pin_D5;
"SRAM_Hi/data-SRAM_Lo/data[7]" : LOCATION = Pin_G9;
"SRAM_Hi/data-SRAM_Lo/data[8]" : LOCATION = Pin_F8;
"SRAM_Hi/data-SRAM_Lo/data[9]" : LOCATION = Pin_E8;
SRAM_Lo/cs_n : LOCATION = Pin_E4;
SRAM_Lo/oe_n : LOCATION = Pin_A2;
SRAM_Lo/we_n : LOCATION = Pin_E6;
```

Block Diagram:

The CPU interfaces with the external FLASH, seven segment LED display, the UART debug interface, the button interface and the LCD display see figure 2 for the details.

Figure 2: CPU Block diagram



Sorting Algorithm – source code:

//Initialization:

```
#include <stdio.h>
#define arr_size 10
#include "pio_lcd16207.h"
#include "nios.h"

// prototyping the functions
void selectionSort(int numbers[], int array_size);
void bubbleSort(int numbers[], int array_size);

void printArray(int nos[], int array_size, int scnt);
void outPutToLCD (int numbers[],int array_size, int swap_cnt, char []);

// global variable
int   oNum[]={3,2,4,1,7,8,6,9,5,0};
char  bsort[]={'B','S'};
char  ssort[]={'S','S'};
```

//Main module:

```
int main()
{
  int choice=0;
  int i,as;

  int arrNum[]={3,2,4,1,7,8,6,9,5,0};

  as= arr_size;

  nr_pio_lcdinit(na_lcd_pio); // initialize the lcd to display

  for(i=0;i<arr_size;i++)
    arrNum[i] = oNum[i];
  printf("Unsorted Array\n");
  printArray(arrNum,arr_size,choice);
  printf("Sorting using Bubble Sort\n");
  bubbleSort(arrNum,as);

  for(i=0;i<arr_size;i++)
    arrNum[i] = oNum[i];
  printf("Unsorted Array\n");
  printArray(arrNum,arr_size,choice);
  printf("Sorting using Selection Sort\n");
  selectionSort(arrNum,as);
}//main
```

//Selection Sort module:

```
void selectionSort(int numbers[], int array_size)
{
  int swpCnt=0;
  int i=0;
  int j;

  int min, temp;
  //  printArray(numbers,array_size,i);
  for (i = 0; i <= array_size-1; i++)
    {
      min = i;
      for (j = i+1; j < array_size; j++)
      {
        if (numbers[j] < numbers[min])
          min = j;
      }
      swpCnt++;
      temp = numbers[i];
      numbers[i] = numbers[min];
      numbers[min] = temp;
      printArray(numbers,array_size,swpCnt);
      outPutToLCD(numbers,array_size,swpCnt,ssort);
      nr_delay(1400);
    }
}
```

// Bubble Sort module:

```
void bubbleSort(int numbers[], int array_size)
{
  int swpCnt=0;
  int i=0;
  int j, temp;
  //  printArray(numbers,array_size,i);
  for (i = (array_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;
            swpCnt++;
            printArray(numbers,array_size,swpCnt);
            outPutToLCD(numbers,array_size,swpCnt,bsort);
            nr_delay(1400);
          }
      }
    }
}
```

//LCD Write Routine:

```
void outPutToLCD (int n[],int asize, int swp, char type[])
{ // form the 32 character string
  char str[128];

sprintf(str,"arr%c%c:%d%d%d%d%d%d%d%d%d%dswp%d:%d%d%d%d%d%d%d%d%d%d",type[
0],type[1],
oNum[0],oNum[1],oNum[2],oNum[3],oNum[4],oNum[5],oNum[6],oNum[7],oNum[8],oN
um[9], swp,n[0],n[1],n[2],n[3],n[4],n[5],n[6],n[7],n[8],n[9]);
      nr_pio_lcdwritescreen(str);
}
```

//Print through UART to stdout:

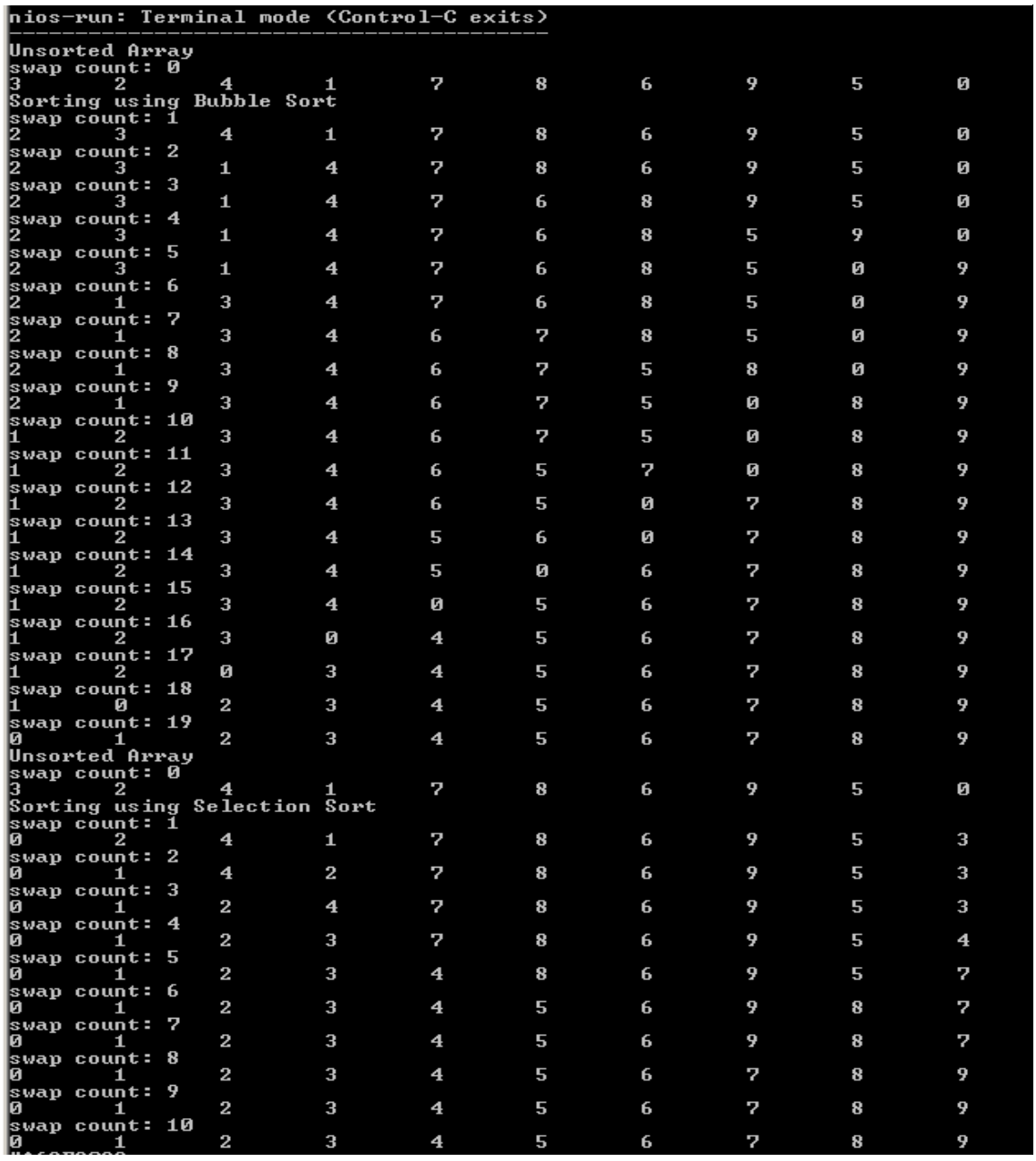
```
void printArray(int nos[], int a_size, int scnt)
{
  int i;
  printf("swap count: %d\n",scnt);
  for(i =0;i<a_size;i++)
    printf("%d\t",nos[i]);

  printf("\n");
}
```

Results:

The simulation results in figure 3 confirm the performance comparison of the 2 sorting algorithms, bubble sort sorted the given array of numbers{3,2,4,1,7,8,6,9,5,0} in ascending order in 19 swaps while the Selection sort did the same in 10 swaps.

The sequence sort needs ‘n’ swaps in the worst case for sorting ‘n’ numbers. At every pass it finds the minimum value of the series and stores it in the first index and increments the index of the minimum to the next location. Bubble sort in comparison, compares adjacent elements for every pass and hence needs ‘(n-1)!’ swaps in the worst case for sorting ‘n’ sequence numbers.

Figure 3: Sorting algorithm output



Conclusion:

The labs 4 & 5 taught us to create the Nios softcore processor and enabled us to code in ‘C’ and run our programs in the 32 bit processor.

The bubble sort and selection sort algorithms were implemented and the sorting procedure was displayed in the 16X2 character LCD screen.

Design of a Keypad User Interface for logon and to change temperature/light settings

LAB #6 EE 2160
November 1, 2004

BY:
Narayanan Krishnamurthy

Table of Contents

Purpose:

The objective of lab6 was to design and implement a keypad based user interface using the excalibur target board and nios 32 bit processor. The LCD and the seven segment LED act as the output interface to the user, prompting him with appropriate messages and displaying the pressed digits.

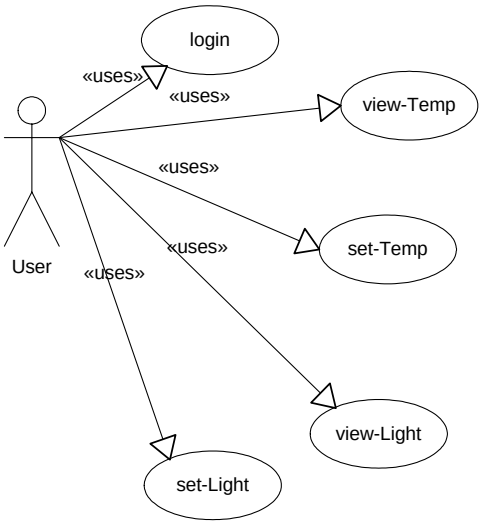
Requirements:

The user should be prompted for a two digit password, and his passcode has to be validated to gain access into the system. The validated user has options to view and change temperature and light settings, the display should prompt to enable the user to do the same, and display the results of his action.

Use Case:

The user interactions with the system can be depicted using a Use-Case, see Figure.1. The user thus can login to the system and has provisions to view and change temperature and light settings as indicated in the use IF.

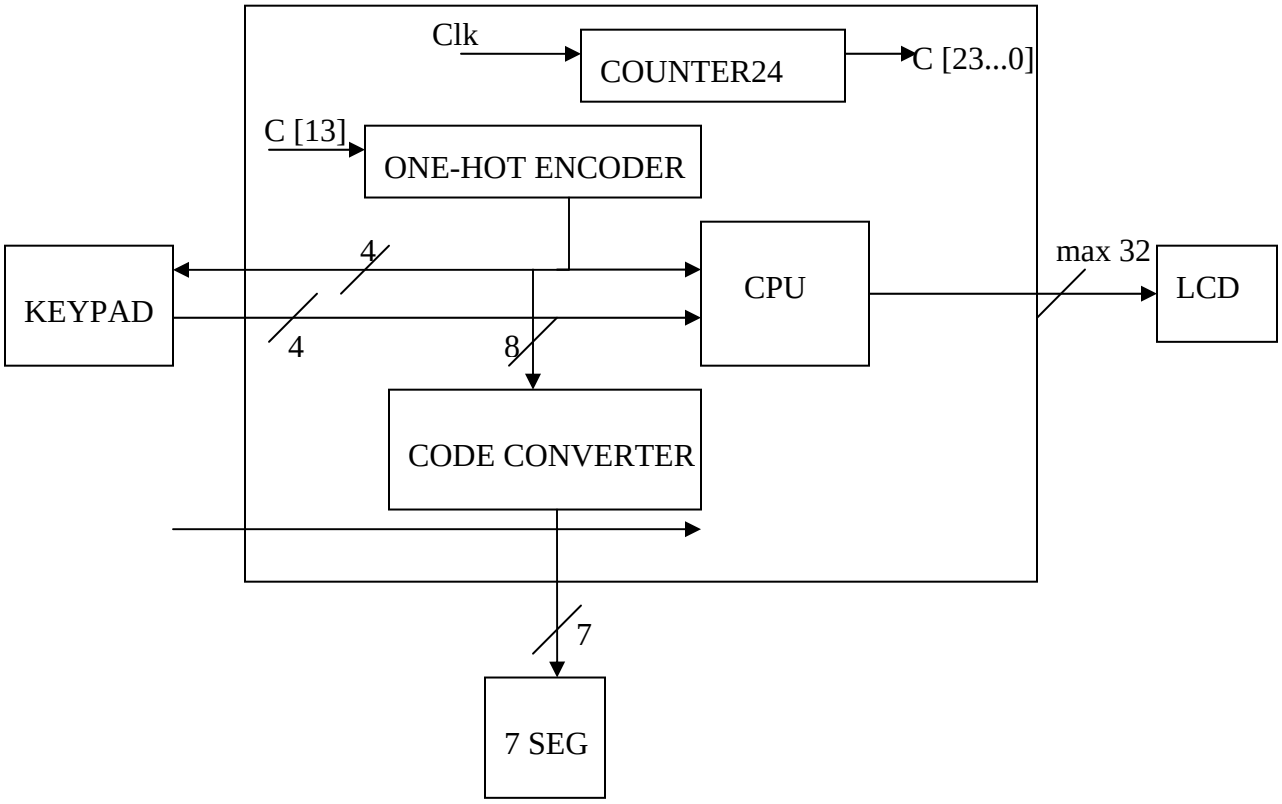
Figure1: Use Case for Keypad Interface system



Specifications:

The first step in the design process was to obtain a top level block diagram. The main components of the block diagram were: 24 bit counter, one-hot encoder, keypad, CPU, code converter, 7 segment display, and LCD. Four bit row was input from the one-hot encoder to the keypad and to the CPU. One hot encoder was stepped down to ensure sufficient number of instructions can be executed in the ISR. Keypad's column output was input to the CPU. The combined signal from the one-hot encoder and the keypad was input to the 7 segment display. The LCD was interfaced with the CPU.

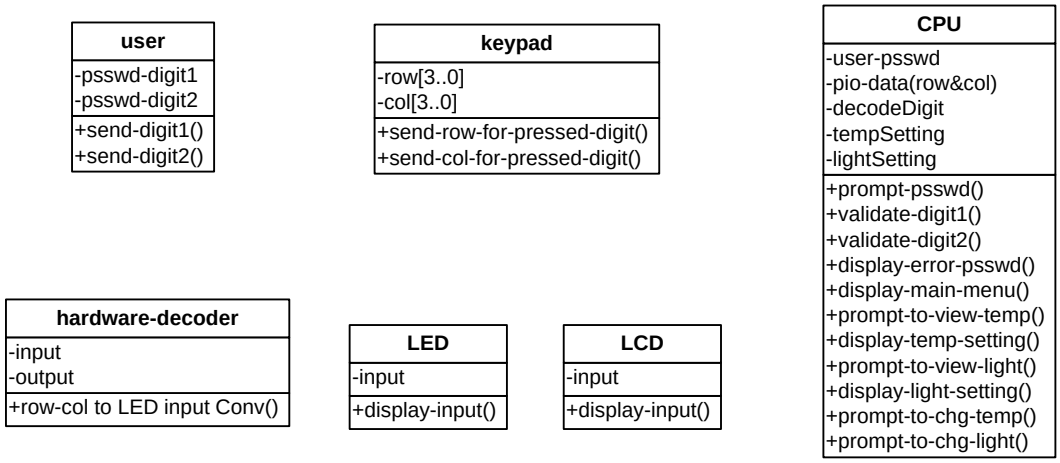
Top level block diagram



UML class diagram:

Class diagrams are useful in identifying the different objects that interact in the system. A class is a characterisation of a collection of objects with similar attributes and behaviour. Thus a class identifies the properties that an object would inherit when it is instantiated. An object(inherits from the class) stores attributes and posseses methods that can change the attributes when interacting with other objects in the system. See Figure 2 for the system class diagram.

Figure2: Class diagram for the user-interface-system

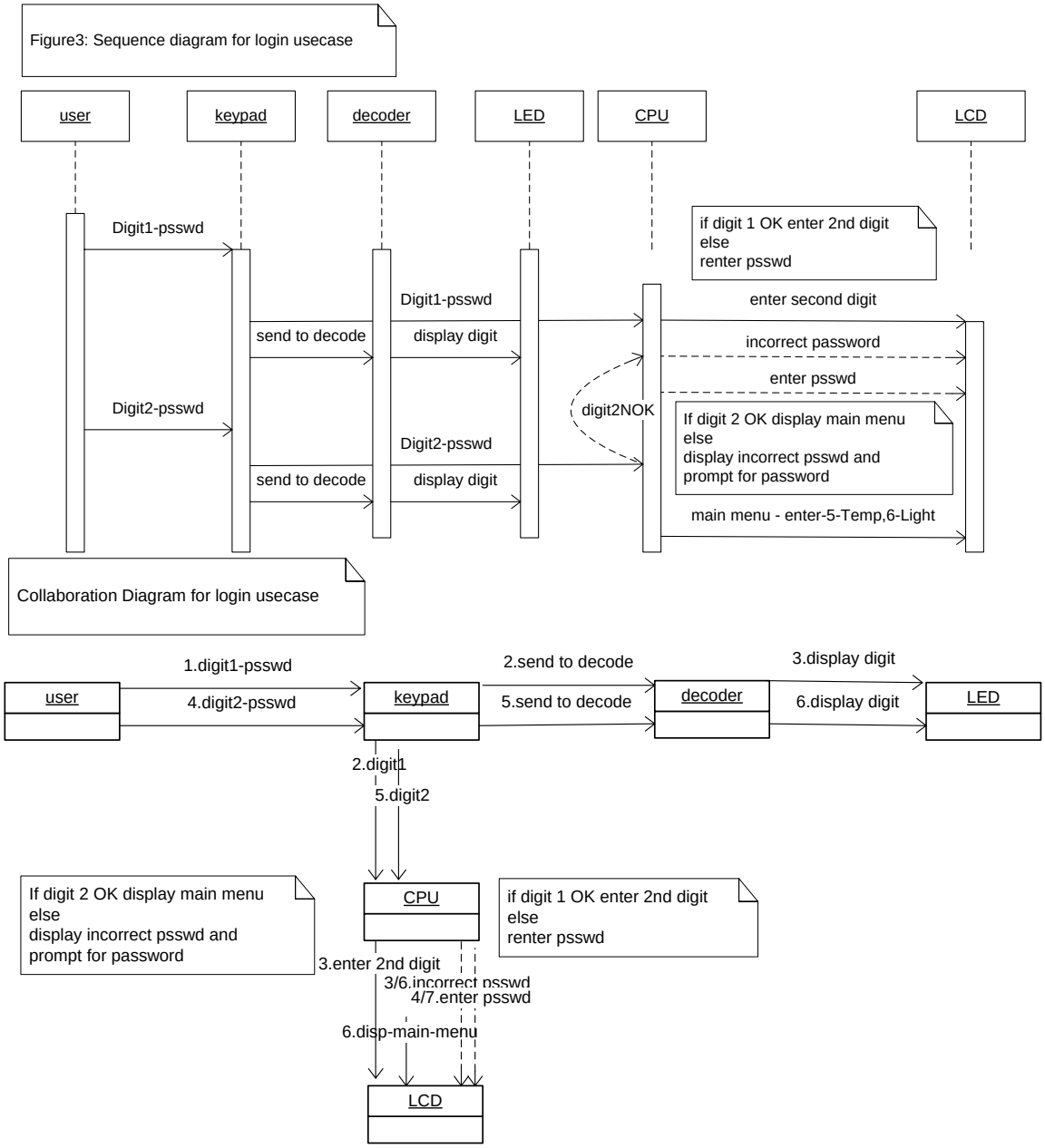


UML Sequence & Collaboration diagrams:

A Sequence diagram enables us to understand the messages transacted between different objects in the system. It gives the control flow and sequence of messages between objects for each use-case.

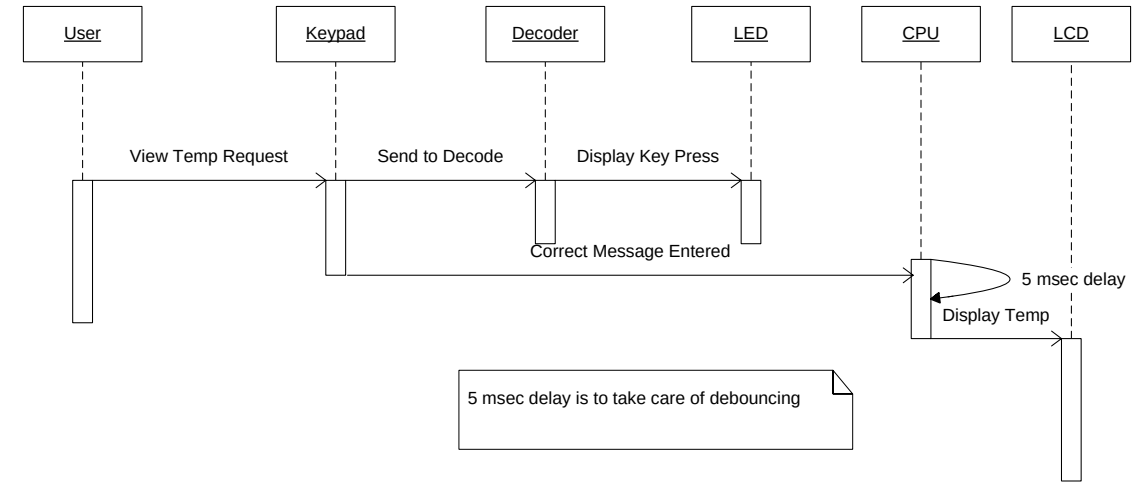
Collaboration diagrams are similar to sequence diagrams in depicting the messages and the sequence in which they are transacted at the object level.

Sequence and Collaboration diagram for: LOGON

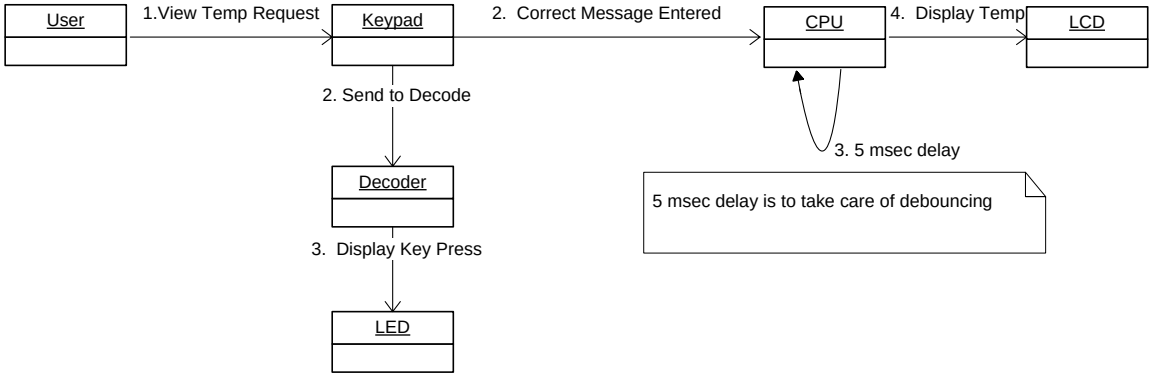


Sequence and Collaboration diagram for: VIEWING TEMPERATURE

Figure4: Sequence Diagram for view-Temp use case



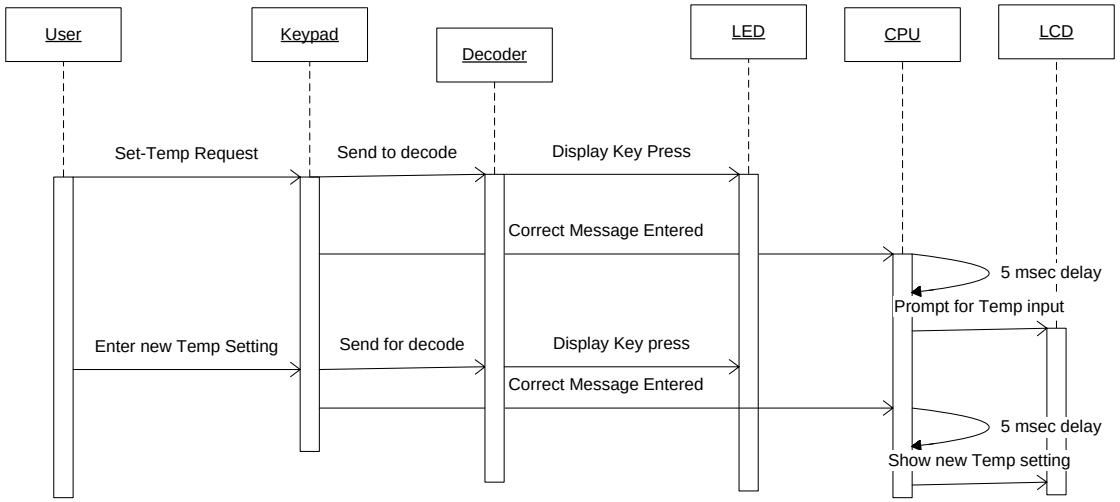
Collaboration Diagram for view-Temp use case



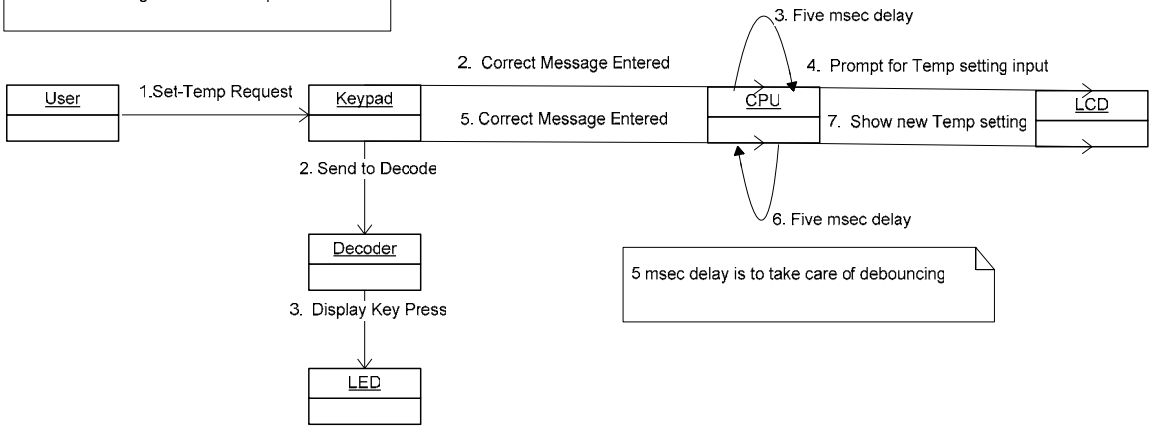
Sequence and Collaboration diagram for: SETTING TEMPERATURE

Figure5 :Sequence Diagram for set-Temp use case

5 msec delay is to take care of debouncing

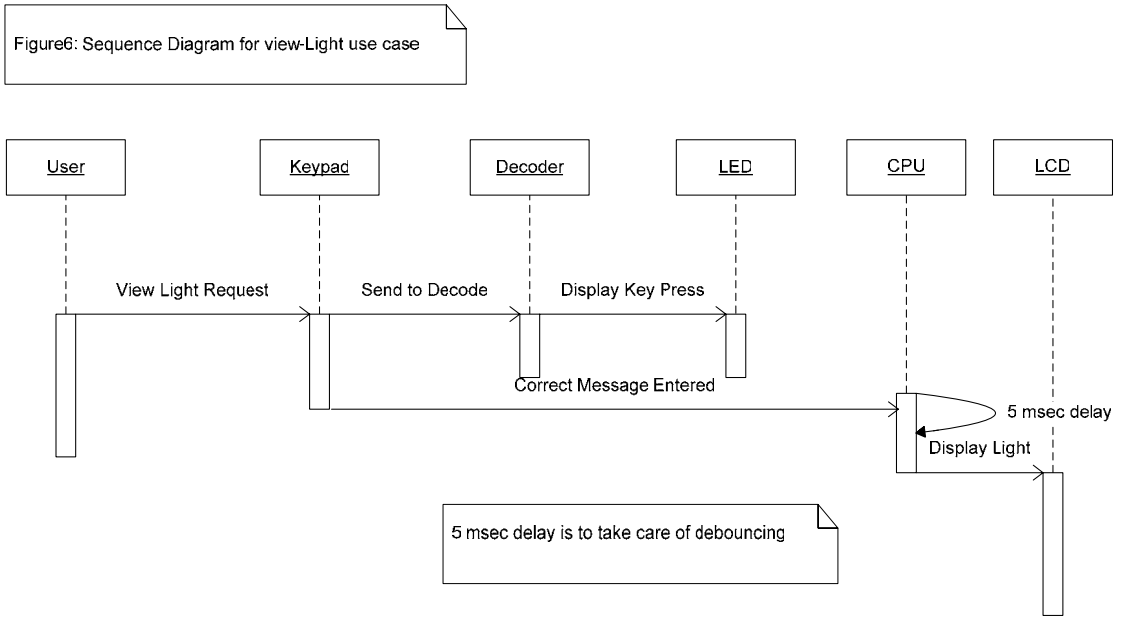


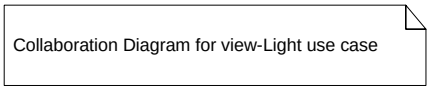
Collaboration Diagram for set-Temp use case

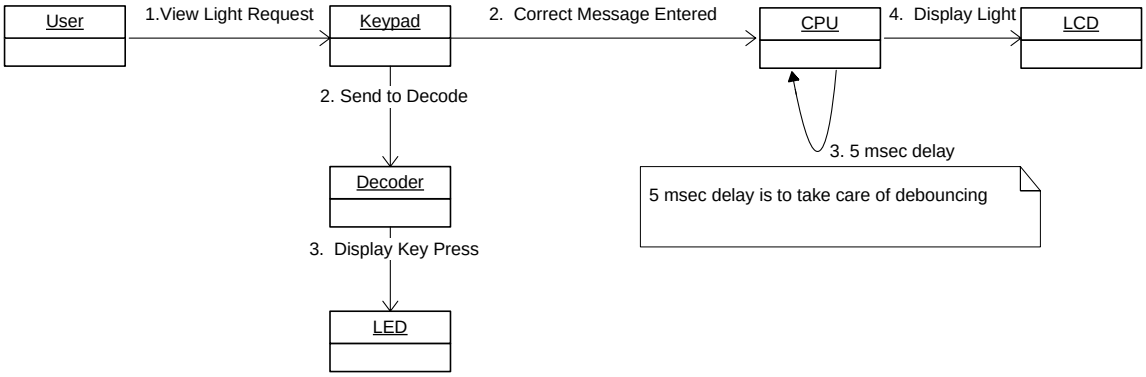


5 msec delay is to take care of debouncing

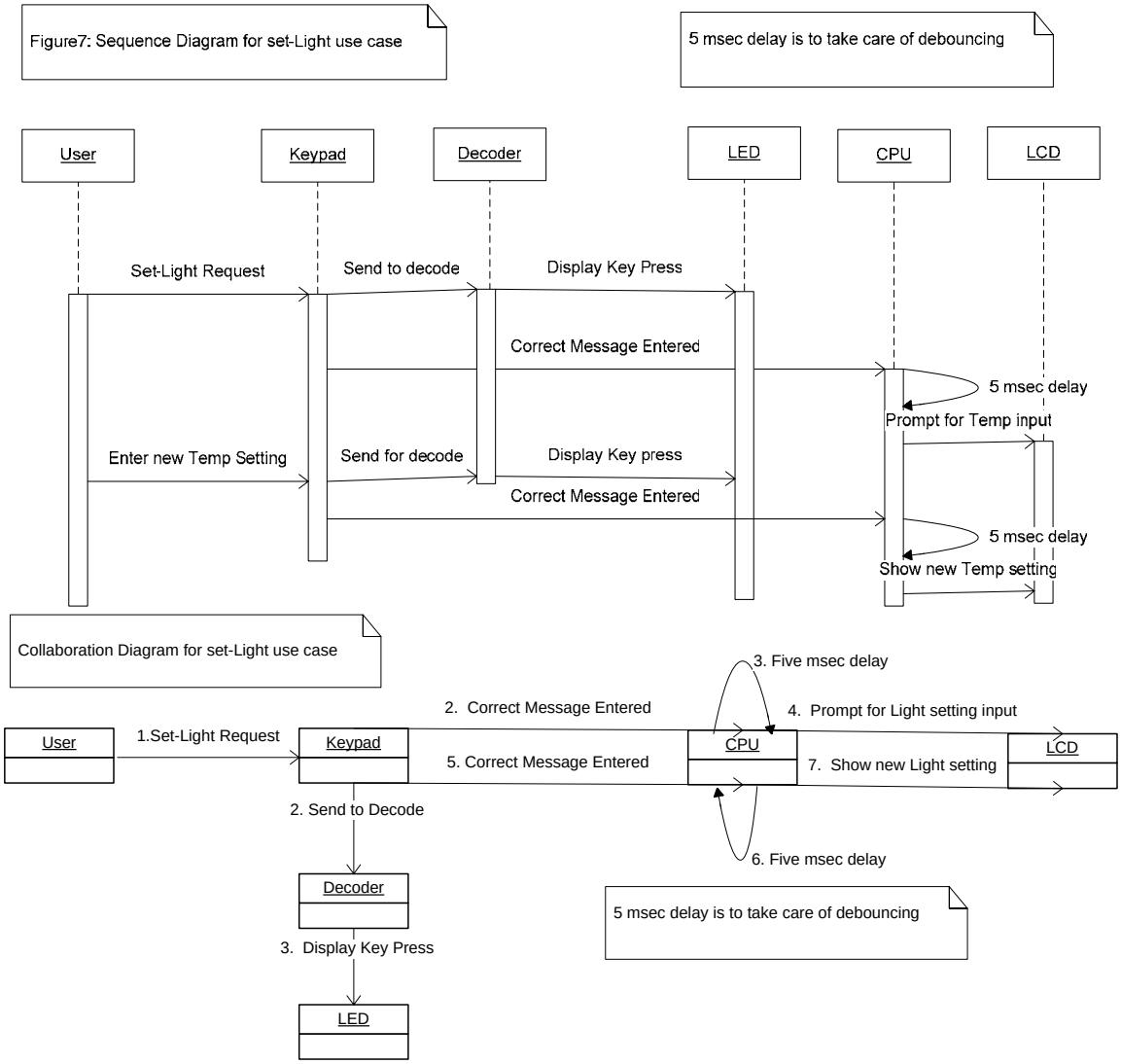
Sequence and Collaboration diagram for: VIEWING LIGHTING







Sequence and Collaboration diagram for: SETTING LIGHTING



Architecture:

The design of the keypad user interface can be decomposed into subcomponents; the first subcomponent should scan the rows and columns of the keypad to detect the digit pressed. The key pad columns are pulled down in our case using 3.9KOhm resistors and the key pad rows are fed by a 4- bit one-hot encoder. The one hot encoder was designed using the following state table Table (1.1) and K-Maps Table(1.2)

Design of one-hot encoder:

Note: One hot encoder was designed with just the clock input, so even if the registers start at any state, at the first clock its reset and the second clock on the counter starts counting like a one-hot counter (See Figure 8 for schematic).

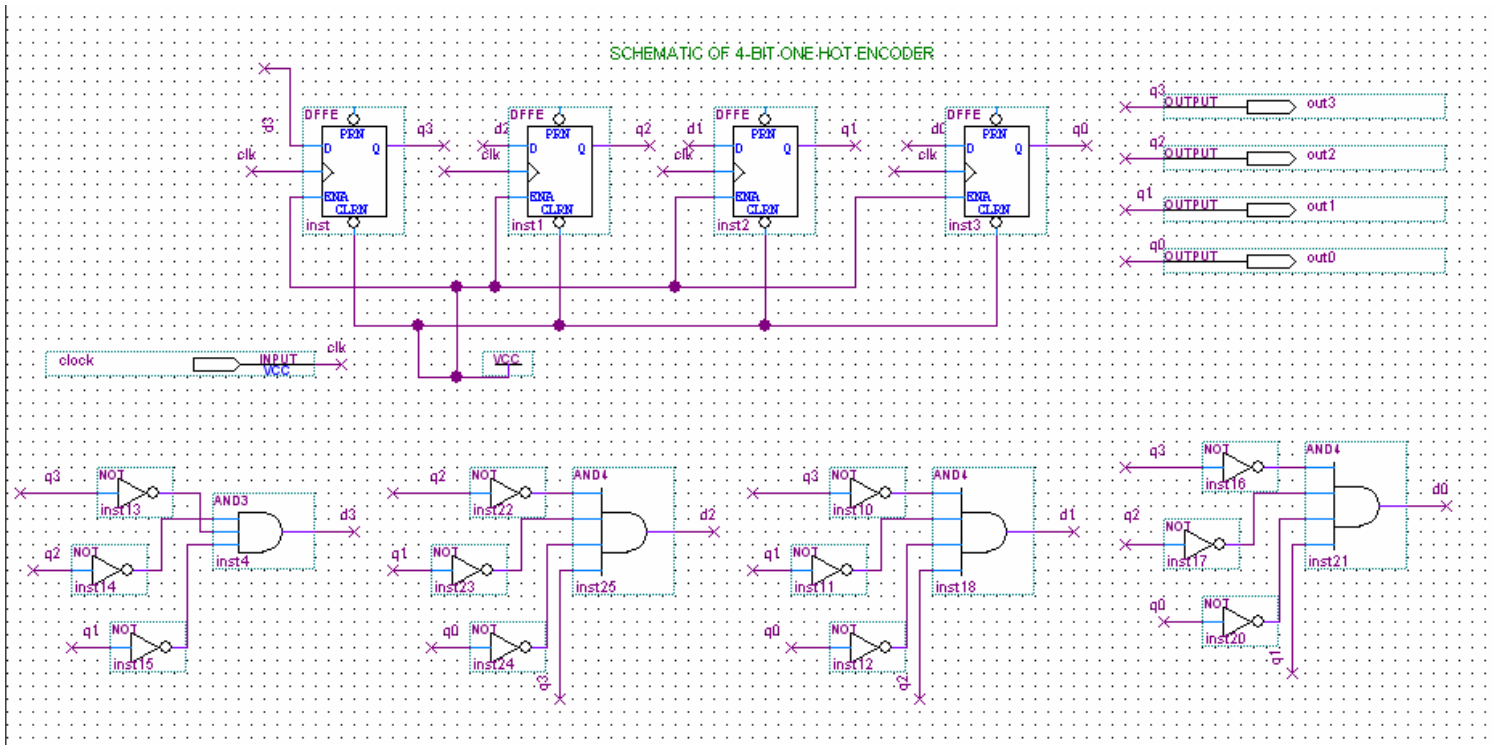


Figure 8. Schematic of One-Hot-Counter

Table 1.1 : Current State -Next State

(Current State) Q3Q2Q1Q0	(Next State) D3D2D1D0
1000	0100
0100	0010
0010	0001
0001	1000
0000	1000
XXXX	0000

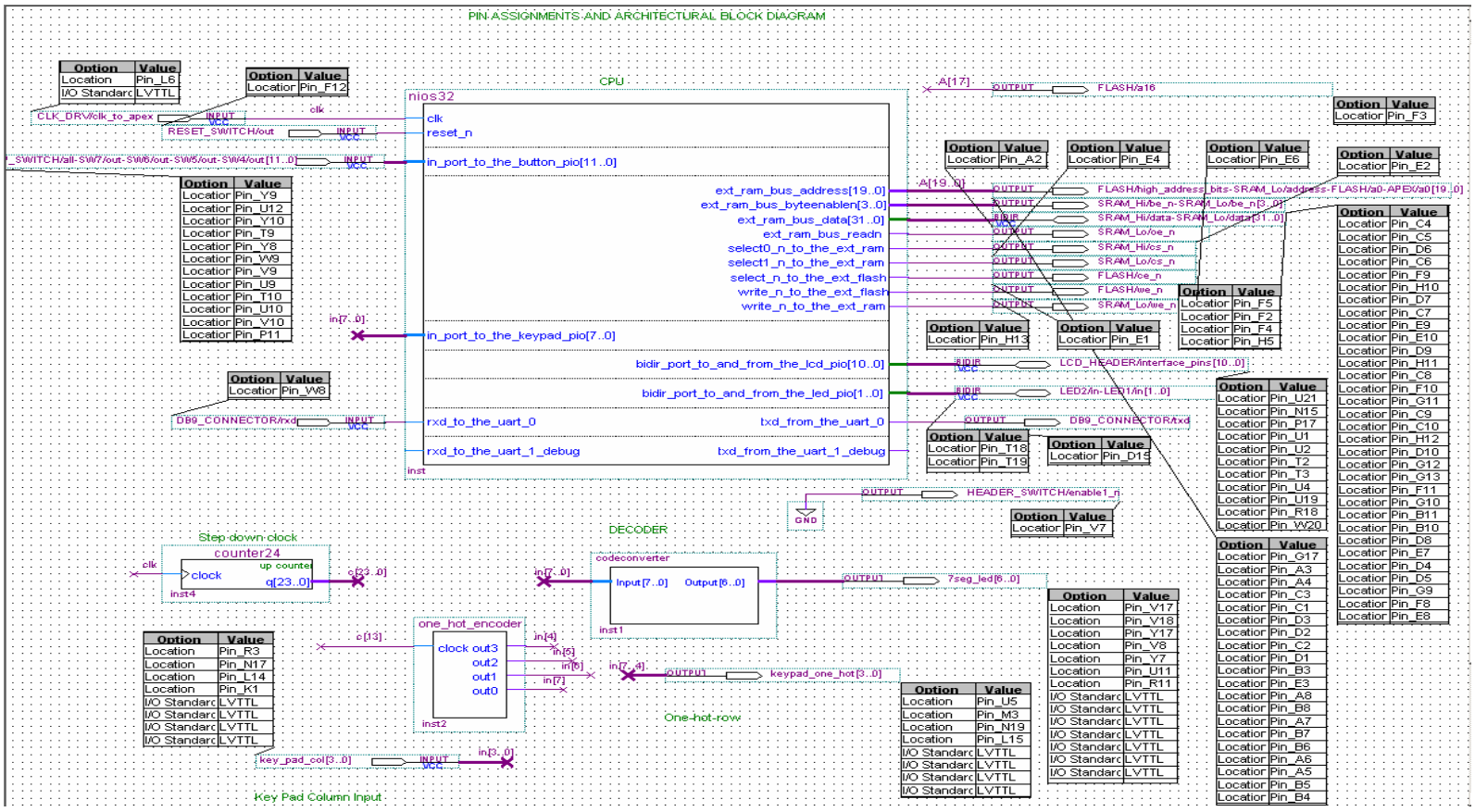
Table 1.2: K-Map for one-hot counter

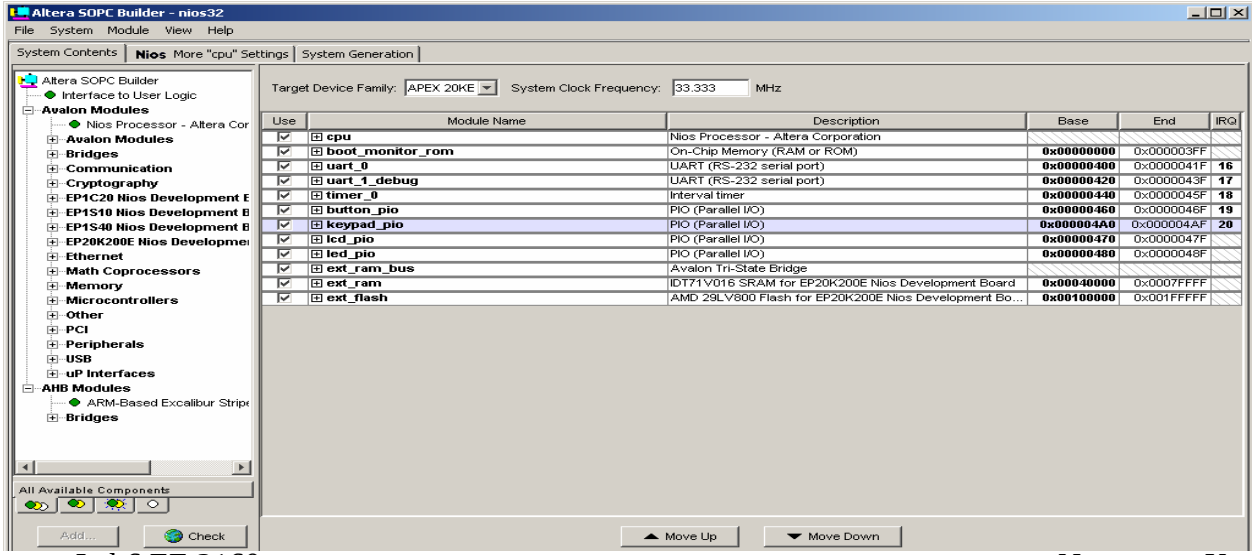
D3		00	01	11	10
	00	1	1	0	0
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0
D2		00	01	11	10
	00	0	0	0	0
	01	0	0	0	0
	11	0	0	0	0
	10	1	0	0	0
D1		00	01	11	10
	00	0	0	0	0
	01	1	0	0	0
	11	0	0	0	0
	10	0	0	0	0
D0		00	01	11	10
	00	0	0	0	1
	01	0	0	0	0
	11	0	0	0	0
	10	0	0	0	0

Architecture Block Diagram

Putting it all together: CPU, One-hot-counter and Decoder

The CPU's interface to seven-segment LED was removed as the rows and columns were directly decoded for the LED's. An 8 Input Port PIO was created using the SOPC builder for the CPU. Debouncing is done in software using delay(5) a 5msec delay before the pio_data is read. Interrupt was masked only for the column values, thus an interrupt is generated only when a key is pressed.

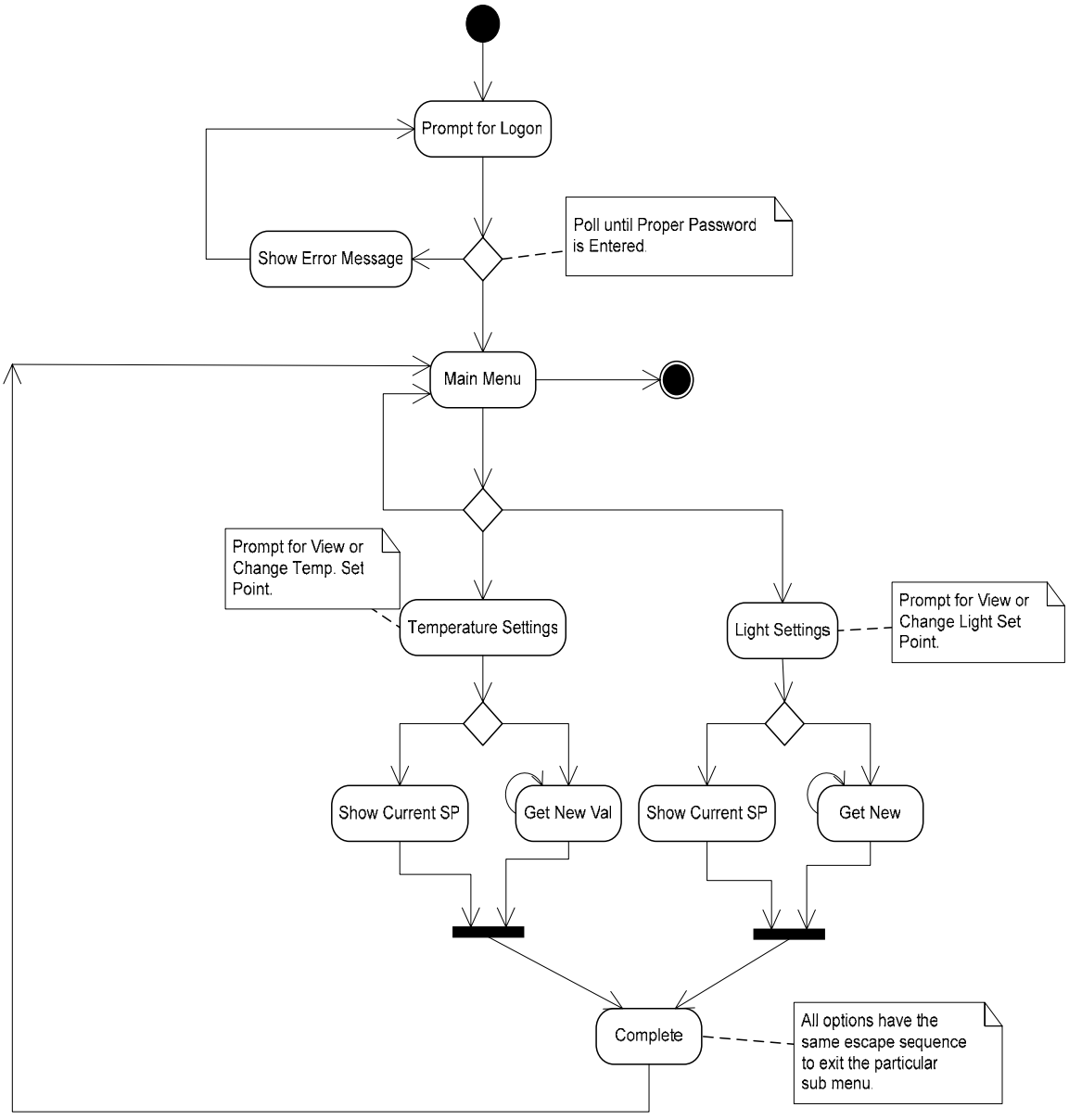




Program state-machine for the keypad interface:

The state machine in C runs in the ISR, the main program is a forever loop that displays the menu appropriate to the current state of the state machine. “lab6.h” has all the #defines for the state machine, decoding the digits. “main.c” implements the user interface and the ISR handler (void keypadPIO_ISR(int context)) the ISR handler in turn calls (int decodeDigit(void)) to decode the received digit. Note: The programming takes care of multiple digit input for the temperature and light settings, but the consecutive digits have to be different for it to be detected. Can be fixed using a timer (to be done in lab-7)

Figure 9 : Program state machine



Pseudo-code of user interface state machine:
"state" global variable stores the states that controls the user interface program running in the softcore nios processor.
The STATES that implement the program state machine (see Figure 9) are:

LOGON	MAIN_TEMP_VIEW
FIRST_OK	MAIN_TEMP_SET
MAIN_MENU	MAIN_LIGHT_VIEW
MAIN_TEMP_MENU	MAIN_LIGHT_SET
MAIN_LIGHT_MENU	

(Refer "lab6.h" for details). The program is initialized to the LOGON state where the user is prompted for password. Two users NK, NG are encoded with a two digit password. The two digit password of the user is validated, if incorrect error message is displayed and user is prompted again. The ISR handler sets the "state" variable to the appropriate state, while the main loop displays the menu appropriate to that state.

Pseudo Code for Main module:

```
//the main program displays the menus depending on the states of the
"state" variable
// check state variable to display menu in 'main' module
      IF LOGON:
        delay(1000); // 1 sec delay
        prompt("Please Enter the Password");


      IF FIRST_OK:
        delay(1000);
        prompt("Please Enter second digit");

      IF MAIN_MENU:
        delay(3000);
        prompt("Press 5 for Temp Press 6for Light");

      IF MAIN_TEMP_MENU:
        delay(1000);
        prompt("Press 7for view, Press 8for Set");

      IF MAIN_TEMP_SET:
        delay(1000);
        prompt("Enter 3digit Temp Setting");

      IF MAIN_LIGHT_MENU:
        delay(1000);
        prompt("Press 3for  view, Press 4for Set");

      IF MAIN_LIGHT_SET:
        delay(1000);
        prompt("Enter 3digit Light Setting");

      ELSE:
        //do nothing
```

Pseudo Code for ISR Handler module:

```
// decodededecodedDigit stores the digit pressed value, check state
// variable to process pressed digit appropriately

IF LOGON
if((decodededecodedDigit == user1_psswd_digit_1)OR
(decodededecodedDigit == user2_psswd_digit_1))
          {
            state = FIRST_OK;
            prompt("Please Enter    second digit\n");
          }
        else // display psswd rejected and prompt again!
          {
            prompt ("Incorrect password try again!");
            state=LOGON;
          }

IF FIRST_OK
        if((decodededecodedDigit == user1_psswd_digit_2) OR
(decodededecodedDigit == user2_passwd_digit_2))
          {
            state = MAIN_MENU;
          }
        else // display psswd rejected and prompt again!
          {
            prompt("Incorrect password try again!");
            state=LOGON;
          }

IF MAIN_MENU
        if(decodededecodedDigit == TEMP_MENU)
          {
            state = MAIN_TEMP_MENU;
            prompt("Enter 7-to view,8-to change Temp\n");
          }
        else if (decodededecodedDigit == LIGHT_MENU)
           {
             state = MAIN_LIGHT_MENU;
             prompt("Enter 3-to view,4-to change light\n");
           }
        else
    state=MAIN_MENU; // invalid entry takes you back to the main menu
```

```
IF MAIN_TEMP_MENU:
        if(decodededecodedDigit == TEMP_VIEW)
          { // int tempSetting[3] array stores the current setting
            display temperature setting
            state = MAIN_MENU;
            prompt("Press 5 for Temp Press 6for Light\n");
          }
        else if(decodededecodedDigit == TEMP_SET)
          {
            state = MAIN_TEMP_SET;
            prompt("Enter 3 digit Temp Setting\n");  // NOTE THE
PROGRAM WOULD NOT ALLOW IDENTICAL INPUTS,i.e all digits have to be
different
          }
IF MAIN_TEMP_SET
      {
          Accumulate the 3 digits in tempSetting variable
          Display the new temp setting
          state = MAIN_MENU;
          prompt("Press 5 for Temp Press 6for Light\n");
      }

IF MAIN_LIGHT_MENU:
          if(decodededecodedDigit == LIGHT_VIEW)
          { // int lightSetting[3] array stores the current setting
            display light setting
            state = MAIN_MENU;
            prompt("Press 5 for Temp Press 6for Light\n");

          }
        else if(decodededecodedDigit == LIGHT_SET)
          state = MAIN_LIGHT_SET;
        prompt ("Enter 3digit Light Setting\n");  // NOTE THE PROGRAM
WOULD NOT ALLOW IDENTICAL INPUTS,i.e all digits have to be different

IF MAIN_LIGHT_SET:
      {
          Accumulate the 3 digits in lightSetting variable
          Display the new light setting
          state = MAIN_MENU;
          prompt("Press 5 for Temp Press 6for Light\n");

      }
```

Pseudo Code for Decoding of Digit module:

```
Decoding Digit based on PIO_DATA → row and column values
 IF 00010001
   decodedDigit = 1;

 IF 00010010
   decodedDigit = 2;

 IF 00010100
   decodedDigit = 3;

 IF 00100001
   decodedDigit = 4;

 IF 00100010
   decodedDigit = 5;

 IF 00100100
   decodedDigit = 6;

 IF 01000001
   decodedDigit = 7;

 IF 01000010
   decodedDigit = 8;

 IF 01000100
   decodedDigit = 9;

 IF 10000010
   decodedDigit = 0;

 ELSE
   decodedDigit = 0xE;

Return decodedDigit
```

Validation & Test Plan:

The following tasks were performed to test and validate the implementation of the design. Please note that the state diagram provided in the lab handout was used to validate the design.

1. Continuity test was performed on the keypad using multimeter to check the soldering on wire connections.
2. The implementation of one-hot encoder was verified using the simulation waveform of Quartus.
3. Row and column relation was verified using an oscilloscope to verify the column going high when corresponding row was input.
4. Key detection on keypad was done to verify mapping of keys.
5. Keypad's reaction to keeping a key pressed was checked for debouncing
6. Keypad's reaction to pressing a key repeatedly was done to verify the implementation of the one-hot encoder.
7. Implementation of logon state was verified.
8. Password checking was verified.
9. Implementation of main menu was verified.
10. Viewing/changing of temperature setting was verified.
11. Viewing/changing of light setting was verified.
12. Exiting the program using the escape key.
13. The stability of the program was checked by performing various menu selections and providing various inputs for prolonged amount of time.

Implementation:

//lab6.h file

```
// States the main program has to execute

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

#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

// 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
```

//main.c file

```
#include<stdio.h>
#include<stdlib.h>
#include"lab6.h"
#include "pio_lcd16207.h"
#include "nios.h"
#include<string.h>

//function prototypes
void keypadPIO_ISR(int context);
int decodeDigit(void);

//declare global variables
int state = LOGON;
int pio_data ;

int tempSetting[]  ={0,0,0};  // 1digit temp setting
int lightSetting[] ={0,0,0}; // 1digit light setting
int rcvd_pio = 0;
int digitCnt = 0;

int main()
{
  //setting up our keypad_pio isr
  np_pio *pio_pointer = na_keypad_pio;
  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
  pio_pointer->np_piointerruptmask = PIO_DATA_COL_MASK; // use
colum bits for interrupt generation
  // init LCD display
  printf("Please Enter    the Password\n");
  nr_pio_lcdinit(na_lcd_pio);
  while(nr_uart_rxchar(0) != 27) // until ESC key...
    {
      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:
       nr_delay(1000);
       break;
     }
    }
}

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;
}

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_lcdwritescreen("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))
    {
      printf("the state is:%d\n",state);
```

```
      rcvd_pio = pio_data;
      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)
          { 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,"%d%d%d",*ptr++,*ptr++,*ptr);
            nr_pio_lcdwritescreen(strDigit);

            state = MAIN_MENU;
               printf("Press 5 for Temp Press 6for Light\n");
          }
        else if(decodedDigit == TEMP_SET)
          {
            state = MAIN_TEMP_SET;
            printf("Enter 3 digit Temp Setting\n");  // NOTE THE
PROGRAM WOULD NOT ALLOW IDENTICAL INPUTS,i.e all digits have to
be different
          }
        break;
      case MAIN_TEMP_SET:
          tempSetting[digitCnt++] = decodedDigit;
          if(digitCnt == NO_DIGITS_IN_SETTING)
          {
            digitCnt=0; ptr = &tempSetting[0];
            printf("Temp Setting is
%d%d%d\n",*ptr++,*ptr++,*ptr);
            strDigit[0] ='\0'; ptr = &tempSetting[0];
            sprintf(strDigit,"%d%d%d",*ptr++,*ptr++,*ptr);
            nr_pio_lcdwritescreen(strDigit);

            state = MAIN_MENU;
            printf("Press 5 for Temp Press 6for Light\n");
          }
        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,"%d%d%d",*ptr++,*ptr++,*ptr);
            nr_pio_lcdwritescreen(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");  // NOTE THE
PROGRAM WOULD NOT ALLOW IDENTICAL INPUTS,i.e all digits have to
be different
        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,"%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
}
```

// CodeConvertor.vhd file

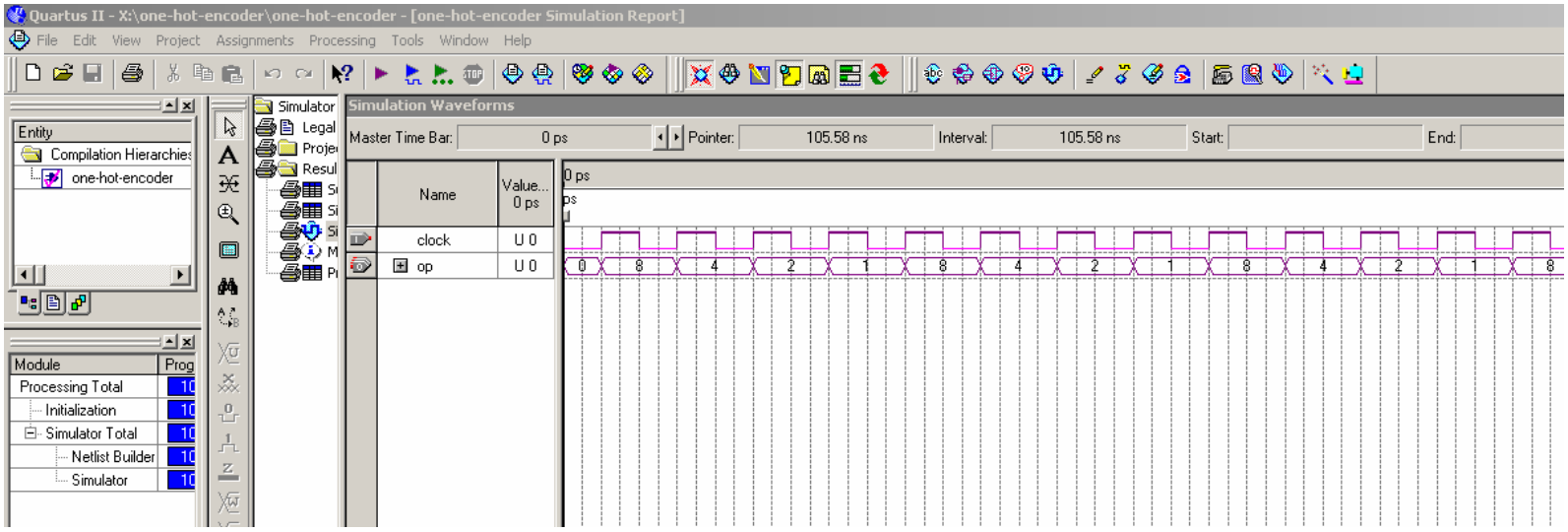
```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;

ENTITY CodeConverter IS
      PORT(
            Input   : IN      std_logic_vector(7 DOWNTO 0);
            Output  : OUT     std_logic_vector(6 DOWNTO 0)
      );
END CodeConverter;


ARCHITECTURE dataflow OF CodeConverter IS
BEGIN
Output <=         "1001111" when Input = "00010001" else
                  "0010010" when Input = "00010010" else
                  "0000110" when Input = "00010100" else
                  "0001000" when Input = "00011000" else
                  "1001100" when Input = "00100001" else
                  "0100100" when Input = "00100010" else
                  "0100000" when Input = "00100100" else
                  "1100000" when Input = "00101000" else
                  "0001111" when Input = "01000001" else
                  "0000000" when Input = "01000010" else
                  "0000100" when Input = "01000100" else
                  "0110001" when Input = "01001000" else
                  "0000001" when Input = "10000010" else
                  "1000010" when Input = "10001000" else
                  "1111111";
END dataflow;
```

Results:

To implement the hardware portion of the design, a one-hot encoder was designed and tested. Please refer to the figure below for the simulation waveform showing the functionality of the implemented one-hot encoder.
After executing the program using Nios SDK shell, the following was seen on the LCD screen:

1. “Please enter the password”: User was prompted to enter his/her password
    a. “Incorrect password try again!”: If the first digit of the password was incorrect, the user was asked to enter the password again.
2. “Please enter the second digit”: If the first digit of the two digit password was correct, the user was prompted to enter the second digit.
    a. “Incorrect password try again!”: If the first digit of the password was incorrect, the user was asked to enter the password again.
3. “Press 5 for Temp Press 6for Light”: If the second digit of the password was correct, the user was presented with the main menu. If the second digit of the password was wrong, then the user was again asked to enter the password.

a. “Press 7for view, Press 8for Set” : If the user pressed 5, he/she was presented with either viewing or setting the temperature setting
    i. “Temp Setting is” : if the user pressed 7, the current temperature setting was shown
    ii. “Enter 3digit Temp Setting”: if the user pressed 8, he/she was asked to enter a 3 digit temperature setting.
b. “Press 3for view, Press 4for Set” : if the user pressed 6, he/she was presented with either viewing for setting the light setting
    i. “Light Setting is” : if the user pressed 3, the current light setting was shown.
    ii. “Enter 3 digit Light Setting” : if the user pressed 4, he/she was asked to enter a 3 digit light setting.



Conclusions:

In this lab assignment, a simple user interface utilizing a keypad, protoboard, and the LCD screen was designed, implemented, and tested. All the prior knowledge was combined into this first application project. One of the problems that were observed was the random key outputs when only one key was pressed. After troubleshooting using a multimeter, a shorted solder contact was seen. Also, after analysis of the JP connector connections, floating ground pins were found that resulted in multiple key detections when the key was pressed just once. The design required a 3 digit setting for the temperature and light. Our design required the 3 digits to be different from each other. We were unable to fix this issue by using the current knowledge of the design. This problem will be fixed from the knowledge gained from lab 7.

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

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).

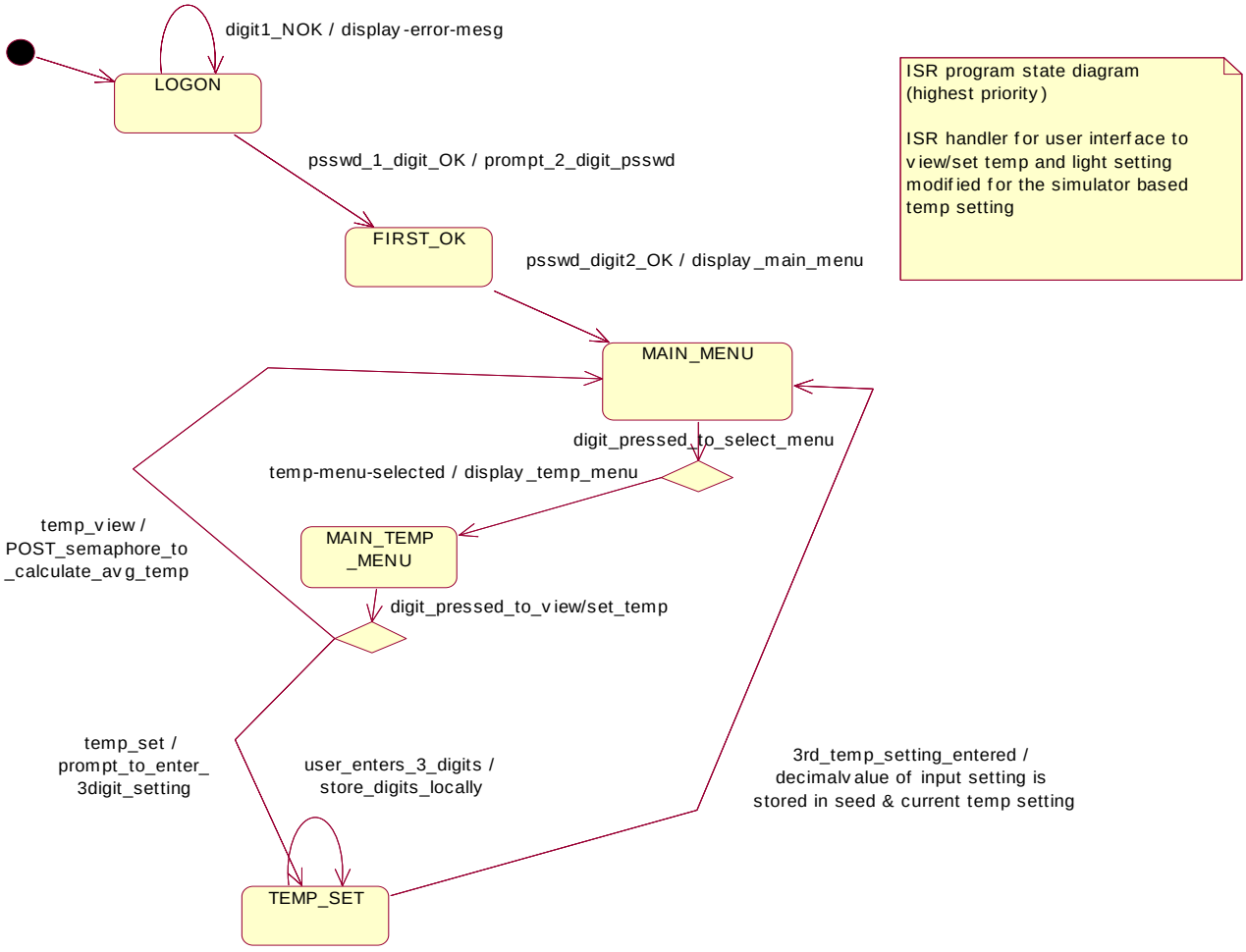
The OSTimeTickHook routine in os_cpu_c.c was used for scheduling the periodic tasks, and for resetting the received pio-data after 150 ticks – 0.75 seconds, this takes care of enabling multiple digit detection based on this threshold.

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

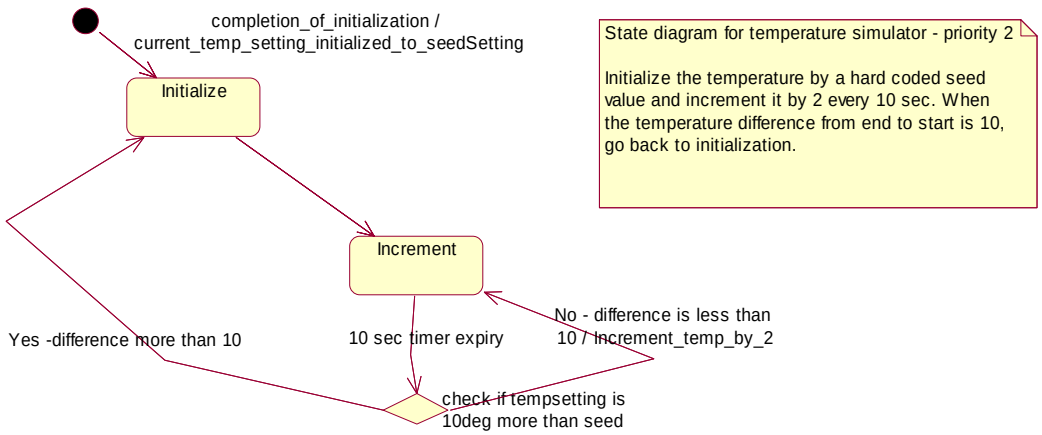
State Diagrams:

ISR Handler –state diagram

Figure 1: State diagram of ISR handler for temp simulator



Temperature simulator – state diagram

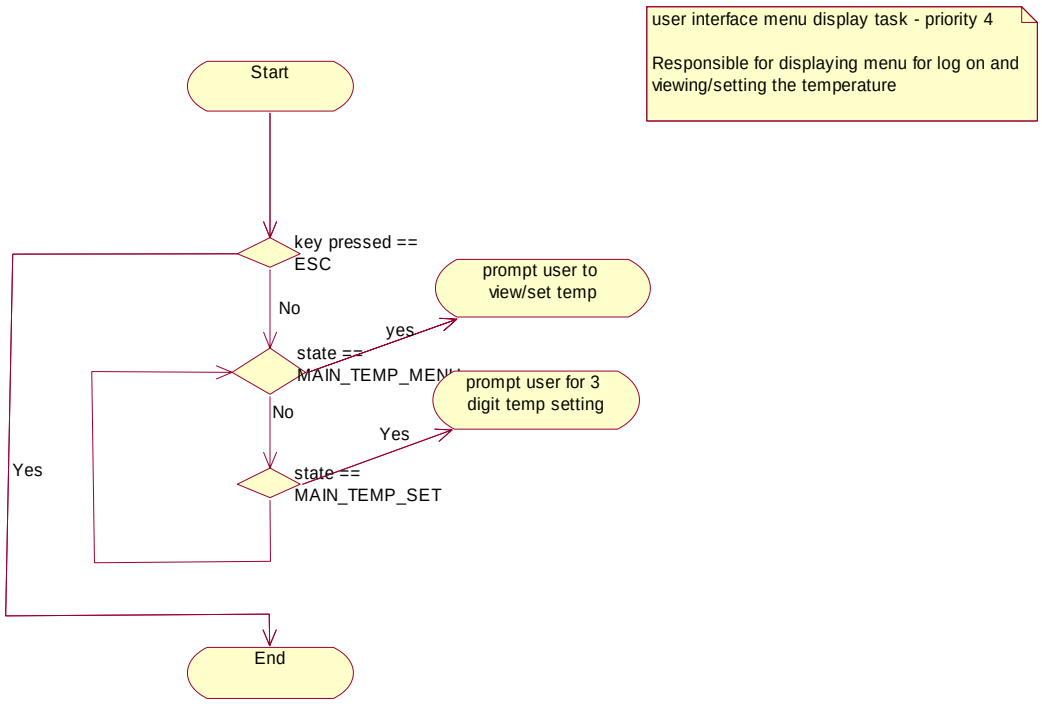


Pseudo-Code:

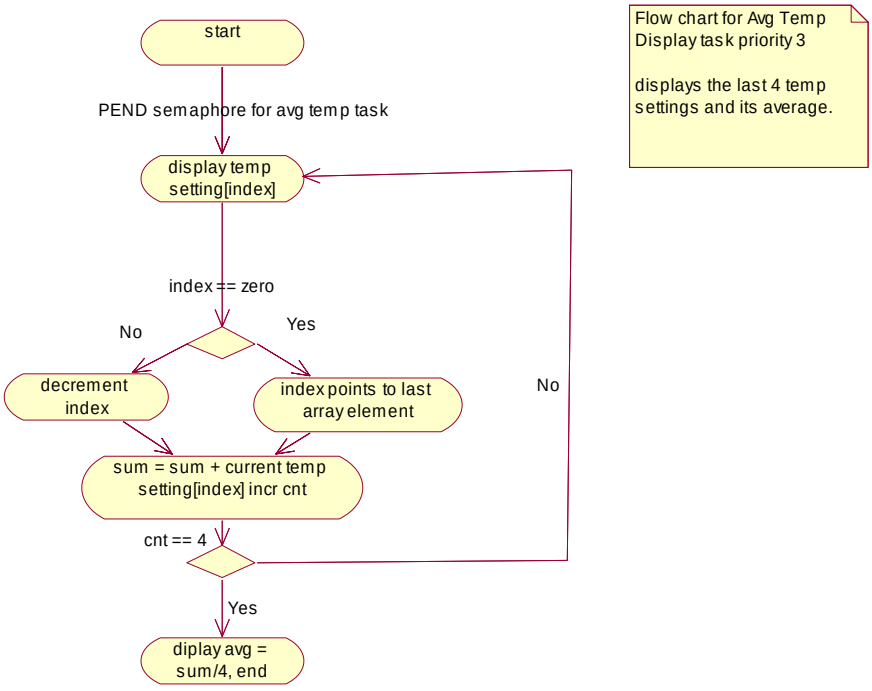
Bubble sort – pseudo code

```
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   //colums 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_lcdwritescreen(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_%d",n[0],n[1],n[2],n[3],n[4],n[5],n[6],n[7
],n[8],n[9]);
      nr_pio_lcdwritescreen(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_lcdwritescreen(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_lcdwritescreen("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)
          {
#ifndef 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");
#if 0
```

```
            /* 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;
#ifndef 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_lcdwritescreen(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_lcdwritescreen(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");
#if 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

```
 [SOPC Builder]$ nb os_cpu_a.s os_cpu_c.c ucos_ii.c multitask.c
--------------------
Beginning Build
--------------------
Sources:
    os_cpu_a.s
    os_cpu_c.c
    ucos_ii.c
    multitask.c

# 2004.11.11 10:24:38 (*) nios-elf-as  --defsym __nios32__=1 --defsym __timeStam
p__=43463 --gstabs  -I .. -I ../.. -I ../inc -I ../../inc -I ../../../inc -I ../
../../../inc -I ../../../../../inc -m32 os_cpu_a.s -o os_cpu_a.s.o

# 2004.11.11 10:24:38 (*) nios-elf-gcc  -I .. -I ../.. -I ../inc -I ../../inc -I
 ../../../inc -I ../../../../inc -I ../../../../../inc -W -Wno-multichar -g -mno
-zero-extend -O2 -m32 os_cpu_c.c -o os_cpu_c.c.o -c

# 2004.11.11 10:24:39 (*) nios-elf-gcc  -I .. -I ../.. -I ../inc -I ../../inc -I
 ../../../inc -I ../../../../inc -I ../../../../../inc -W -Wno-multichar -g -mno
-zero-extend -O2 -m32 ucos_ii.c -o ucos_ii.c.o -c

# 2004.11.11 10:24:42 (*) nios-elf-gcc  -I .. -I ../.. -I ../inc -I ../../inc -I
 ../../../inc -I ../../../../inc -I ../../../../../inc -W -Wno-multichar -g -mno
-zero-extend -O2 -m32 multitask.c -o multitask.c.o -c

# 2004.11.11 10:24:43 (*) nios-elf-ld -e _start -u _start -g -T /cygdrive/c/alte
ra/excalibur/sopc_builder/bin/excalibur.ld  ../lib/obj32/nios_jumptostart.s.o os
_cpu_a.s.o os_cpu_c.c.o ucos_ii.c.o multitask.c.o --start-group -l nios32 -l c -
l m -l gcc --end-group -L/cygdrive/c/altera/excalibur/sopc_builder/bin/nios-gnup
ro/nios-elf/lib/m32 -L/cygdrive/c/altera/excalibur/sopc_builder/bin/nios-gnupro/
lib/gcc-lib/nios-elf/2.9-nios-010801-20030520/m32 -L../lib -L../../lib -L../../.
./lib -L../../../../lib -L../../../../../lib -L../inc -L../../inc -L../../../inc
 -L../../../../inc -L../../../../../inc -L. -o multitask.out

# 2004.11.11 10:24:44 (*) nios-elf-objcopy -O srec multitask.out multitask.srec

# 2004.11.11 10:24:46 (*) nios-elf-nm multitask.out | sort > multitask.nm

# 2004.11.11 10:24:47 (*) nios-elf-objdump -d --source multitask.out > multitask
.objdump

--------------------
Finishing Build
--------------------

/cygdrive/x/lab-7/lab7_1/cpu_sdk/src
[SOPC Builder]$ nr multitask.srec

nios-run: Ready to download multitask.srec over COM1: at 115200 bps

nios-run: Downloading.....................................................
```

```
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
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
```

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.