

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

Table of Contents	2
Purpose:.....	3
Requirements:	3
Use Case:	3
Specifications:.....	3
Top level block diagram	4
UML class diagram:.....	4
UML Sequence & Collaboration diagrams:	5
Sequence and Collaboration diagram for: LOGON.....	6
Sequence and Collaboration diagram for: VIEWING TEMPERATURE.....	7
Sequence and Collaboration diagram for: SETTING TEMPERATURE.....	8
Sequence and Collaboration diagram for: VIEWING LIGHTING.....	9
Sequence and Collaboration diagram for: SETTING LIGHTING.....	10
Architecture:	10
Design of one-hot encoder:.....	10
Architecture Block Diagram	12
Putting it all together: CPU, One-hot-counter and Decoder	12
Program state-machine for the keypad interface:	13
Pseudo Code for Main module:	14
Pseudo Code for ISR Handler module:.....	15
Pseudo Code for Decoding of Digit module:.....	17
Validation & Test Plan:	18
Implementation:	19
//lab6.h file	19
//main.c file	20
// CodeConvertor.vhd file	26
Results:.....	26

Purpose:

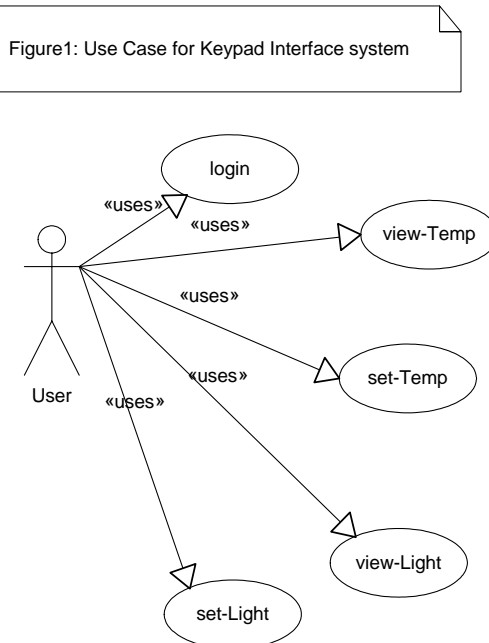
The objective of lab6 was to design and implement a keypad based user interface using the excalibur target board and nios 32 bit processor. The LCD and the seven segment LED act as the output interface to the user, prompting him with appropriate messages and displaying the pressed digits.

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:

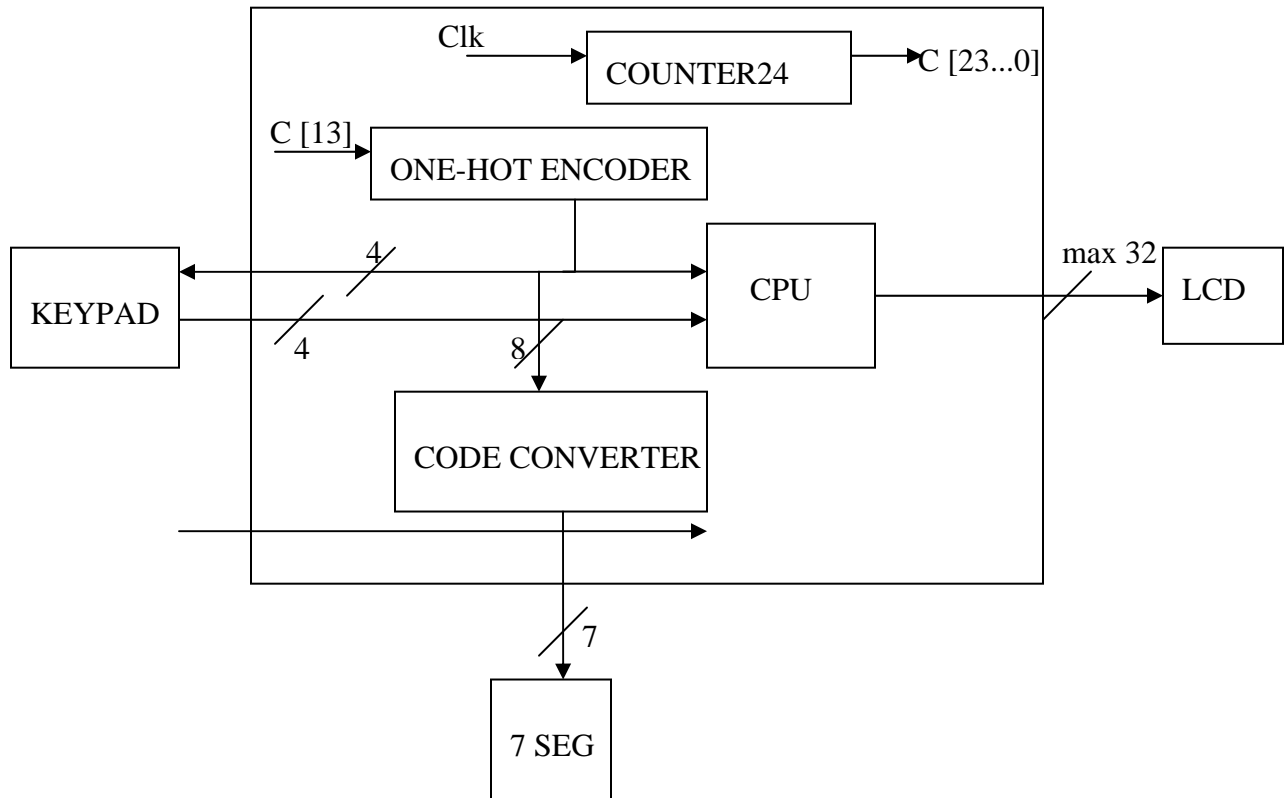
The user interactions with the system can be depicted using a Use-Case, see Figure.1. The user thus can login to the system and has provisions to view and change temperature and light settings as indicated in the use IF.



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.

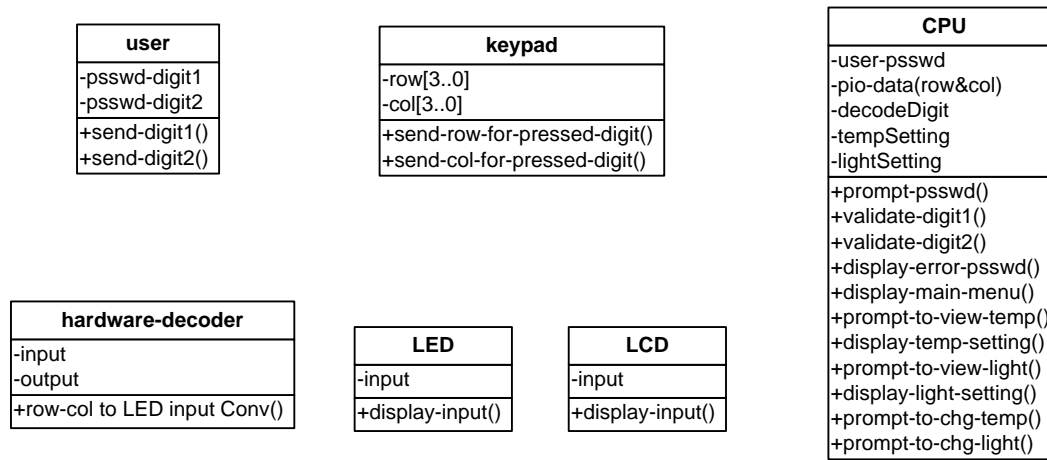
Top level block diagram



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 possesses 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

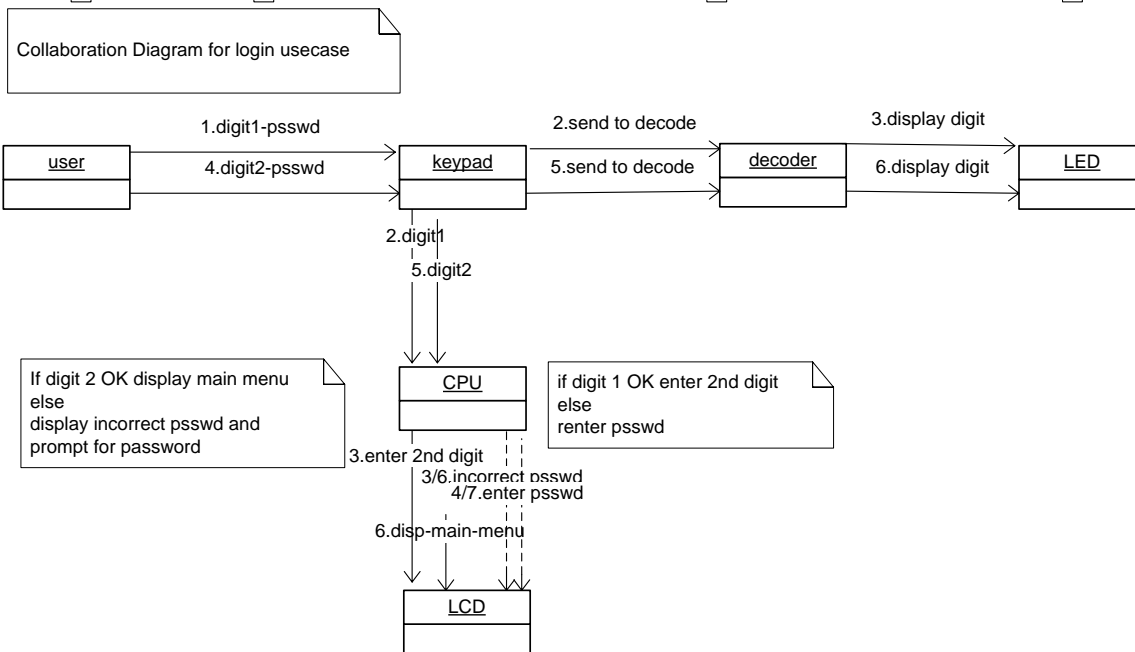
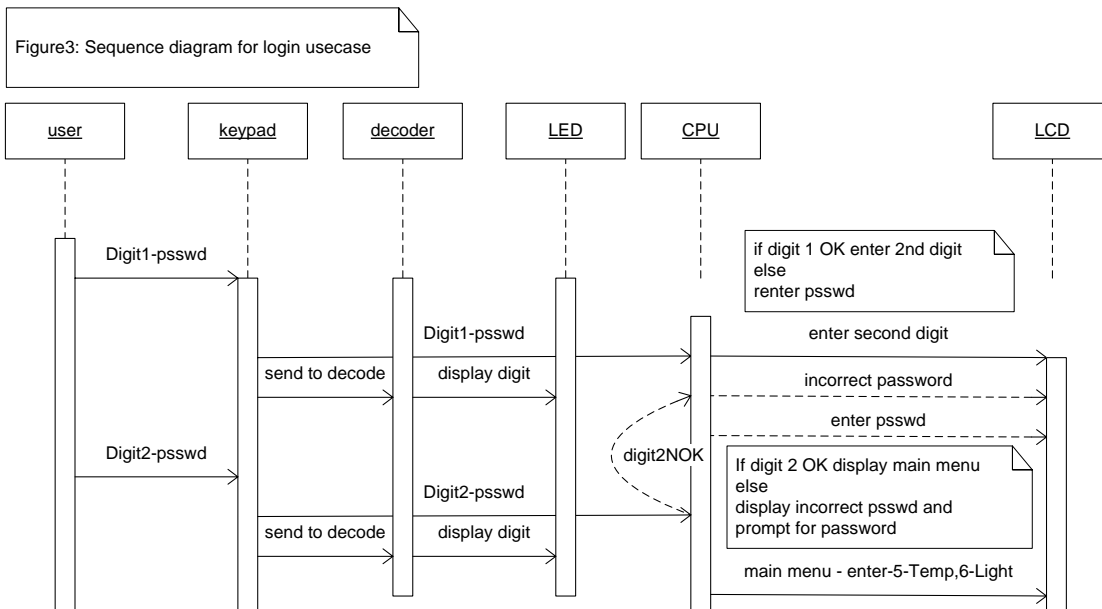


UML Sequence & Collaboration diagrams:

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.

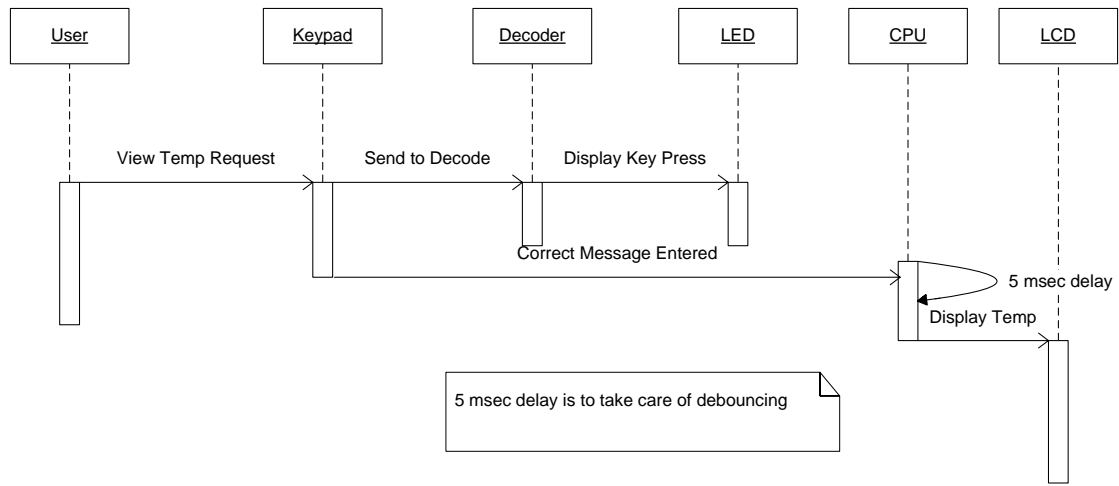
Collaboration diagrams are similar to sequence diagrams in depicting the messages and the sequence in which they are transacted at the object level.

Sequence and Collaboration diagram for: LOGON

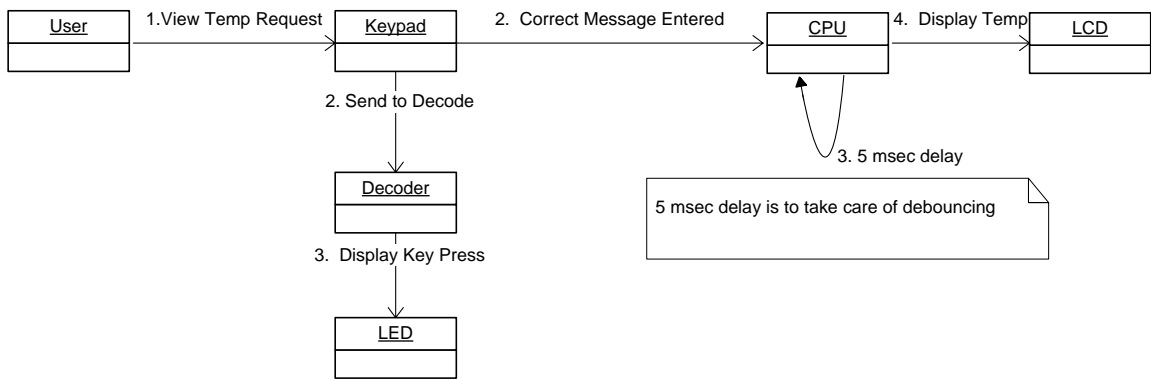


Sequence and Collaboration diagram for: VIEWING TEMPERATURE

Figure4: Sequence Diagram for view-Temp use case



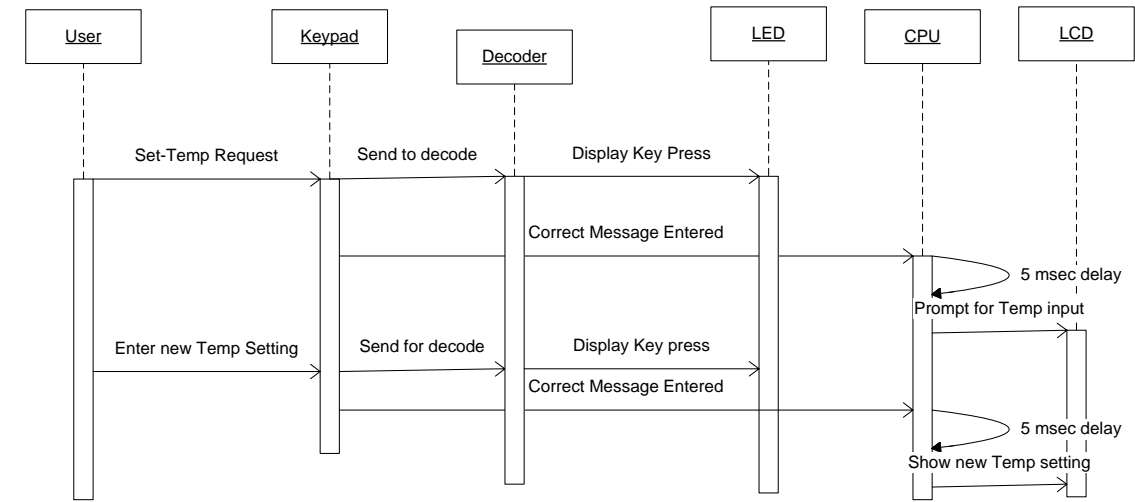
Collaboration Diagram for view-Temp use case



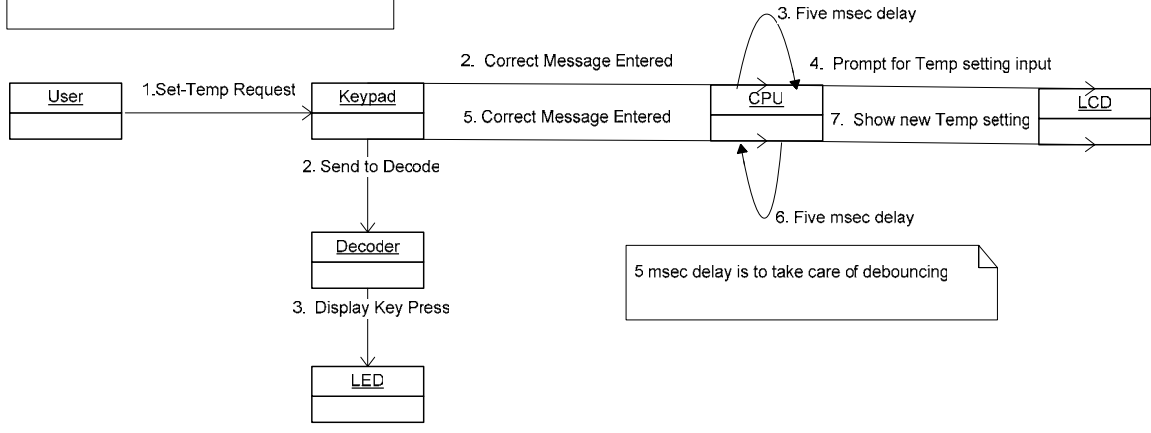
Sequence and Collaboration diagram for: SETTING TEMPERATURE

Figure5 :Sequence Diagram for set-Temp use case

5 msec delay is to take care of debouncing



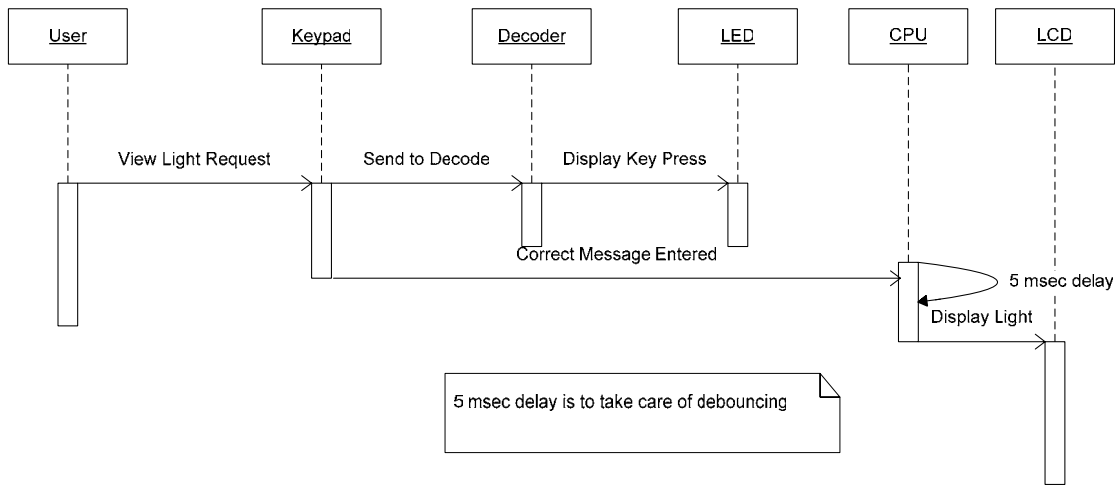
Collaboration Diagram for set-Temp use case



5 msec delay is to take care of debouncing

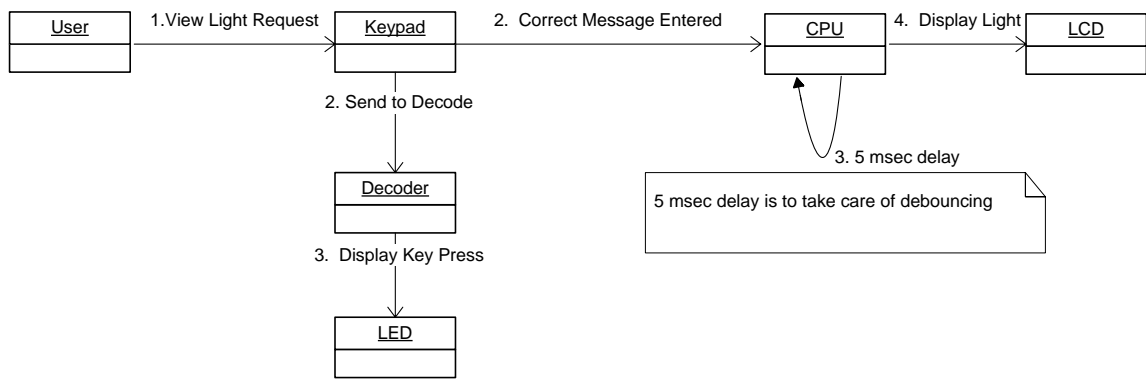
Sequence and Collaboration diagram for: VIEWING LIGHTING

Figure6: Sequence Diagram for view-Light use case



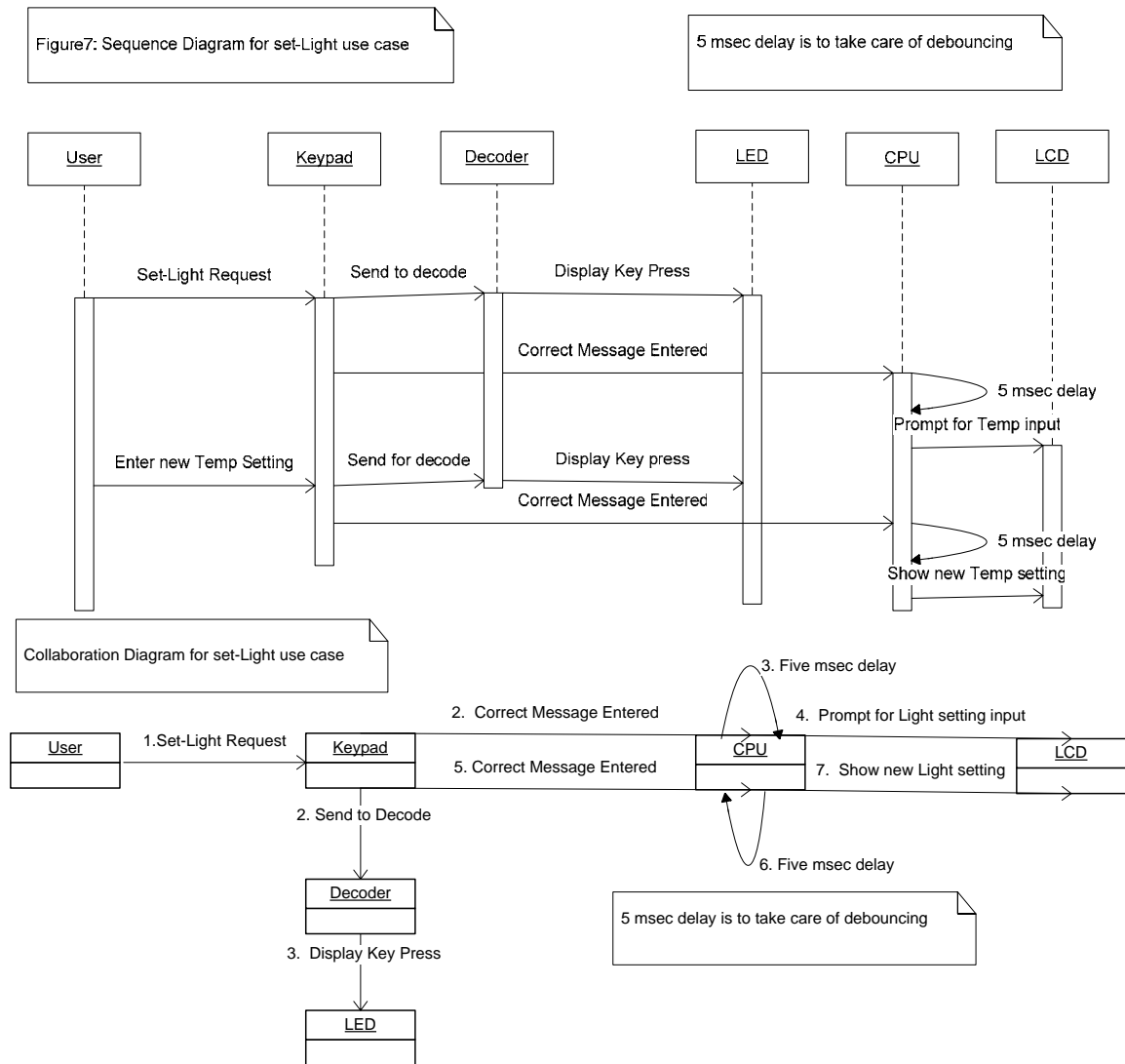
5 msec delay is to take care of debouncing

Collaboration Diagram for view-Light use case



5 msec delay is to take care of debouncing

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).

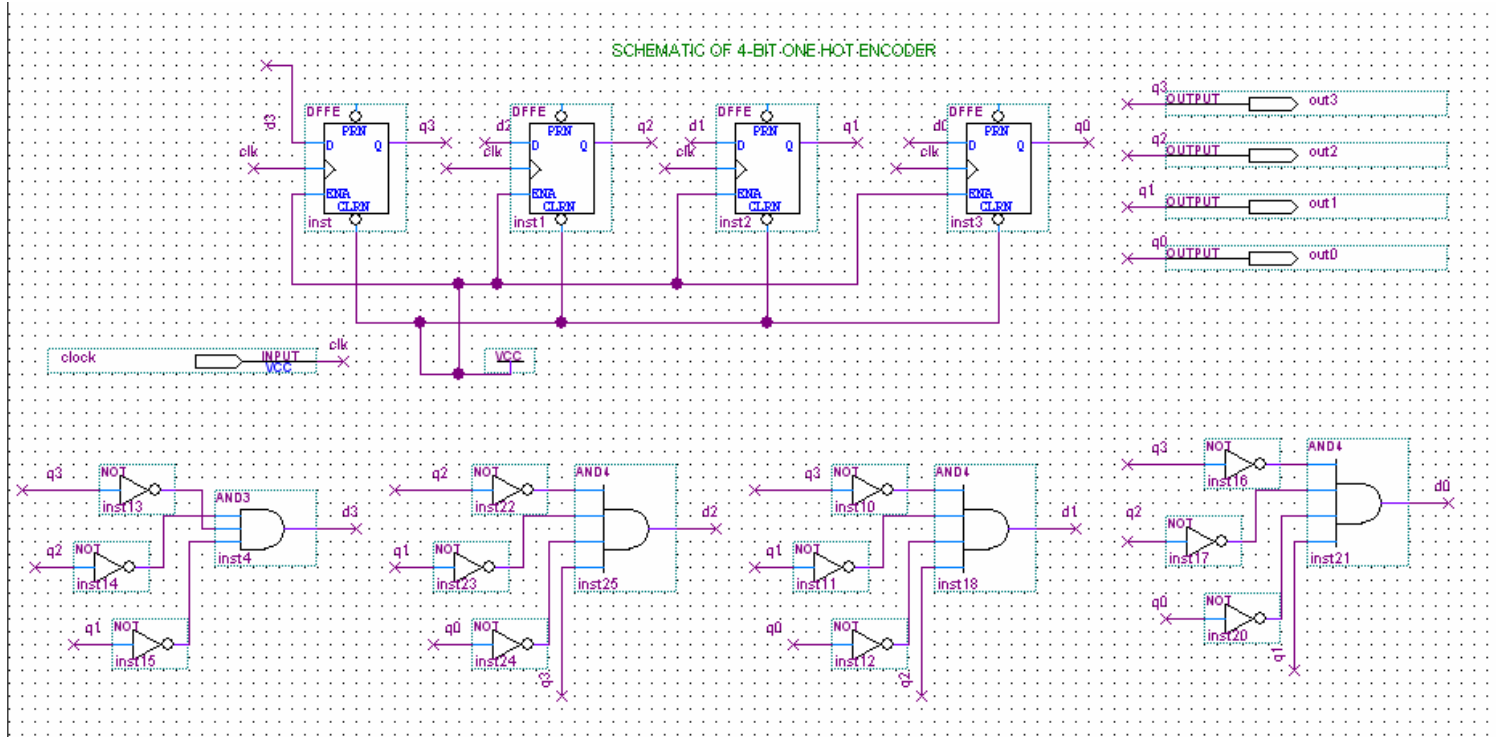


Figure 8. Schematic of One-Hot-Counter

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

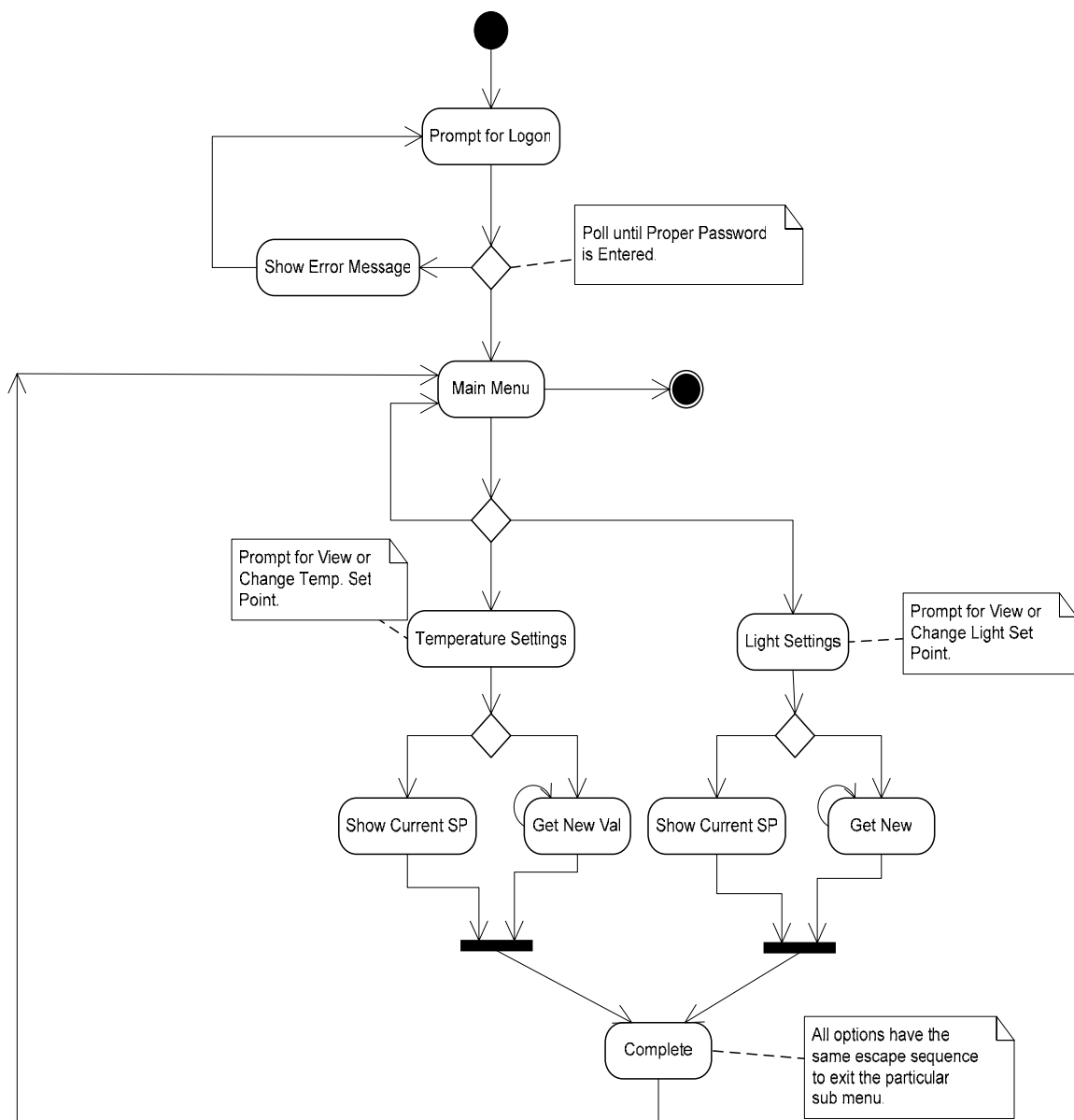
Table 1.2: K-Map for one-hot counter

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

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:

```

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

IF LOGON
if((decodedecodedDigit == user1_psswd_digit_1)OR
(decodedecodedDigit == 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((decodedecodedDigit == user1_psswd_digit_2) OR
(decodedecodedDigit == 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(decodedecodedDigit == TEMP_MENU)
        {
            state = MAIN_TEMP_MENU;
            prompt("Enter 7-to view,8-to change Temp\n");
        }
    else if (decodedecodedDigit == 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(decodeddecodedDigit == 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(decodeddecodedDigit == 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(decodeddecodedDigit == 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(decodeddecodedDigit == 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 //columns 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
column 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;

```

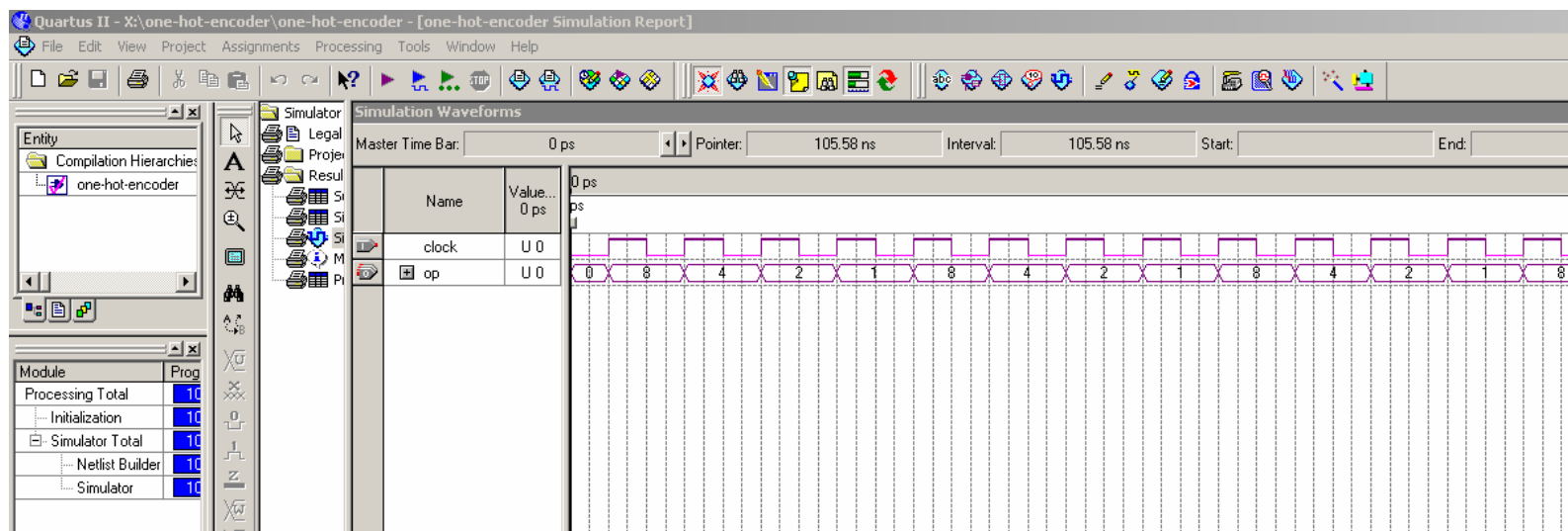
Results:

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.