**EE084IU**

**Micro-processing Systems**

**LAB 3:**

**EXTERNAL INTERUPTS and TIMERS**

**Student: Do Minh Duy**

**Student ID: ITITSB22029**

**Class:**

**Date:**

I. LAB OBJECTIVES

After completion of this Lab, Students will:

* Know how to use External Interrupts
* Interface ATmega32 with LCD to design digital clock
* Know how to use timer and counter.
* Develop multi-tasks program using ATmega32 Microcontroller
* Design the embedded system hardware and software for specific application

II. PROCEDURE

Student Carry out all the Lab tasks and get Lab Instructor Signature check mark for each task. For each Task student should write program in both C Programming and assembly programming.

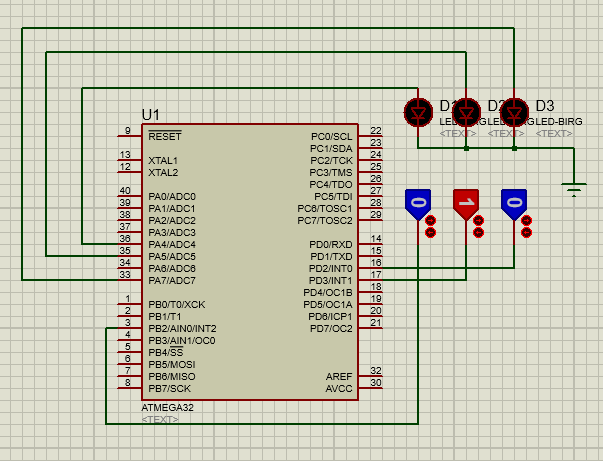
II.1 TASK1 : EXTERNAL INTERUPT INT0, INT1, INT2

II.1.1 Problem Statement:

design a control 3 leds circuit using ATMEGA32 Microcontroller, The microcontroller has input pins to connect to 3 buttons (bt0, bt1, bt2) using INT0, INT1, INT2 , 3 led connected to PA4, PA5, PA7 pins. Write the code to control 3 leds with following requirements :

* Toggle LED0 when INT0 has rising edge trigger event.
* Toggle LED1 when INT1 has falling edge trigger event.
* Toggle LED2 when INT2 has rising edge trigger event.

Write the program in C code and then convert to AVR Assembly.



C code

#include "avr/io.h"

#include "avr/interrupt.h"

int main (){

DDRA = 0xFF; // PA as an output

MCUCR = 0x11; // make INT0 and INT1 rising edge and falling edge triggered respectively

MCUCSR = (1<<ISC2); // make INT2 rising edge triggered

GICR = (1<<INT0)|(1<<INT1)|(1<<INT2);

sei (); // enable interrupts

while (1); // wait here

}

ISR (INT0\_vect){ // ISR for external interrupt 0

PORTA ^= (1<<0) ; // toggle PORTA.0

}

ISR (INT1\_vect){ // ISR for external interrupt 1

PORTA ^= (1<<1) ; // toggle PORTA.1

}

ISR (INT2\_vect){ // ISR for external interrupt 2

PORTA ^= (1<<2) ; // toggle PORTA.2

}

**AVR Assembly**

.include "m32def.inc"

.org 0000 ; reset vector

jmp main

.org 0x0002; External Interrupt request 0

JMP INT0\_ISR

.org 0x0004; External Interrupt request 1

JMP INT1\_ISR

.org 0x0006; External Interrupt request 2

JMP INT2\_ISR

.org 0x2A

main: ; //int main (){

LDI R16,0xFF ; DDRA = 0xFF; // PA as an output

OUT DDRA,R16

LDI R16,0b00001011; MCUCR = 0b00001011; // make INT0 rising edge triggered

OUT MCUCR, R16;

// make INT1 falling edge triggered

LDI R16, (1<<6); MCUCSR = (1<<6); // make INT2 falling edge triggered

OUT MCUCSR, R16;

LDI R16,(1<<INT0)|(1<<INT1)|(1<<INT2); GICR = (1<<INT0)|(1<<INT1)|(1<<INT2) ;

OUT GICR, R16;

// enable external interrupt 0,1,and 2

SEI; sei (); // enable interrupts

while\_1: ;while (1); // wait here

JMP while\_1;

INT0\_ISR: ;ISR (INT0\_vect){ // ISR for external interrupt 0

IN R16,PORTA; PORTA ^= (1<<4) ; // toggle PORTA.4

LDI R17,(1<<4)

EOR R16, R17

OUT PORTA, R16

RETI

INT1\_ISR: ;ISR (INT1\_vect){ // ISR for external interrupt 1

IN R16,PORTA; PORTA ^= (1<<5) ; // toggle PORTA.5

LDI R17,(1<<5)

EOR R16, R17

OUT PORTA, R16

RETI

INT2\_ISR: ;ISR (INT2\_vect){ // ISR for external interrupt 2

IN R16,PORTA; PORTA ^= (1<<7) ; // toggle PORTA.7

LDI R17,(1<<7)

EOR R16, R17

OUT PORTA, R16

RETI

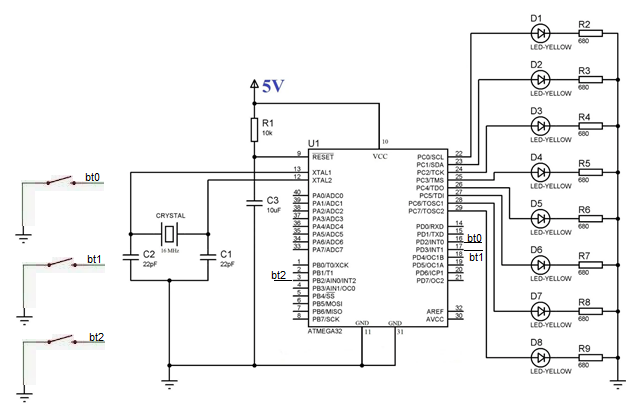
II.2 TASK2 : GPIO TIMER and EXTERNAL INTERUPT

II.2.1 Problem Statement:

design a control 8 leds circuit using ATMEGA32 Microcontroller, The microcontroller has input pins to connect to 3 buttons (bt0, bt1, bt2) using INT0, INT1, INT2 with following requirements

* When bt0 (INT0 working at falling edge) is pressed the 8 leds will toggle in 1 second period
* When bt1(INT1working at falling edge) is pressed the 8 leds will turned on turn by turn : led D0 turned on 1 second and turned off, then led D1 turned on second and turned off… this process will continue to led D7, then the process will repeat again at led D0.
* When bt2 (INT2 working at low level)) is pressed the 8 leds will be turned on gradually (from D0 to D7) with 1 second period for one led, then turned off gradually (from D0 to D7). then the process will repeat again from the beginning.
* Using timer1 over flow to create delay\_1\_second function

(Write the program in C code and Assembly) :



II.2.2 Circuit Design:

Student Design the application circuit in Proteus

II.2.3 Theory:

Student review related theory knowledge about external interrupt

+ Related Registers.

+ External Interrupt INT0, INT1, INT2 setting up.

II.2.4 Algorithm

1. Develop function leds\_control0 to control 8 leds toggled in 1 second period
2. Develop function leds\_control1 to control 8 leds turned on turn by turn.
3. Develop function leds\_control2 to control 8 leds turned on and off gradually
4. Develop delay\_1\_second function using timer1
5. set up INT0 working with falling edge activation mode to set control mode 0
6. set up INT1 working with falling edge activation mode to set control mode 0
7. set up INT0 working with low level activation mode to set control mode 0
8. Write main program

+ Setting up PORTC as output

+ Enable all defined interrupts

+ While(1) loop:

Case 0 : call leds\_control0 function

Case 1 : call leds\_control2 function

Case 2 : call leds\_control2 function

II.2.5 Code

|  |  |
| --- | --- |
| Write general comments about the program here  /\*----------------------------------------------------------------  **Assembly Code**  .include "m32def.inc"  .def i=R18  .org 0000 ; reset vector  jmp main  .org 0x0002; External Interrupt request 0  JMP INT0\_ISR  .org 0x0004; External Interrupt request 1  JMP INT1\_ISR  .org 0x0006; External Interrupt request 2  JMP INT2\_ISR  .org 0x2A  main: ; //int main (){  LDI R16,0xFF ; DDRA = 0xFF; // PA as an output  OUT DDRA,R16  LDI R16,0b00001010; MCUCR = 0b00001011; // make INT0 rising edge triggered  OUT MCUCR, R16;  // make INT1 falling edge triggered  LDI R16, (1<<6); MCUCSR = (1<<6); // make INT2 falling edge triggered  OUT MCUCSR, R16;  LDI R16,(1<<INT0)|(1<<INT1)|(1<<INT2); GICR = (1<<INT0)|(1<<INT1)|(1<<INT2) ;  OUT GICR, R16;  // enable external interrupt 0,1,and 2  SEI; sei (); // enable interrupts  while\_1: ;while (1); // wait here  JMP while\_1;  INT0\_ISR: ;ISR (INT0\_vect){ // ISR for external interrupt 1    LDI R16,0xFF  OUT PORTC,R16  call delay\_1000ms  LDI R16,0x00  OUT PORTC,R16  call delay\_1000ms  RETI    INT1\_ISR: ;ISR (INT1\_vect){ // ISR for external interrupt 1  LDI i,0  LDI R16,0b00000001  For1 : OUT PORTC,R16  call delay\_1000ms  LSL R16  INC i  CPI i,8  BRNE For1  LDI R16,0x00  OUT PORTC,R16  RETI  INT2\_ISR: ;ISR (INT2\_vect){ // ISR for external interrupt 2  LDI i,0  LDI R16,0b10000000  For2 : OUT PORTC,R16  call delay\_1000ms  LSR R16  INC i  CPI i,8  BRNE For2  LDI R16,0x00  OUT PORTC,R16  RETI  delay\_1000ms:  LDI R20,68 ; 1 Cycle  DL3: LDI R21,100 ;  DL2: LDI R22,48 ;  DL1: DEC R22 ;  BRNE DL1 ;  DEC R21 ;  BRNE DL2 ;  DEC R20 ;  BRNE DL3 ;  RET  This program is a multi-tasks program to control 8 leds with different styles  Author: xxxxxxxxx Date: xx.xx.20xx  ---------------------------------------------------------------------\*/ | |
| Instruction | Comments |
|  |  |

II.2.6 Result

Check the designed circuit.

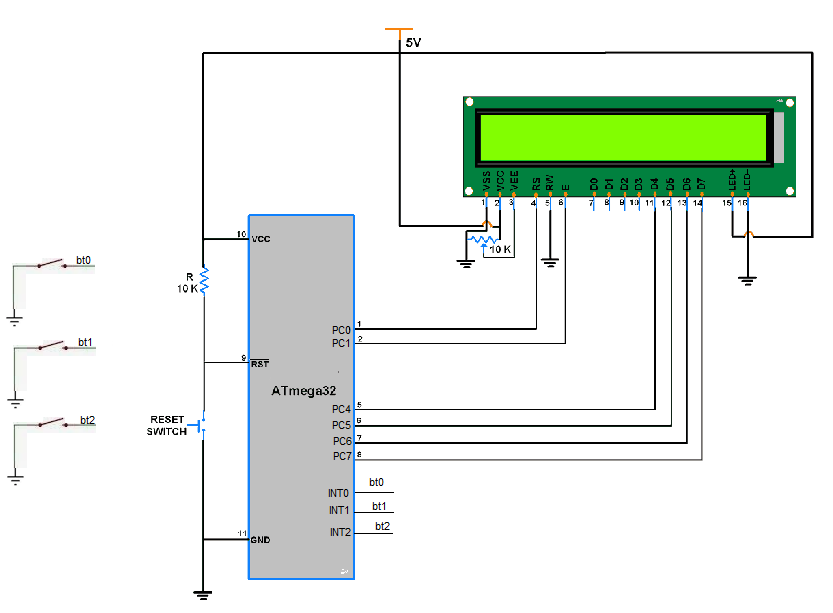
Check Oscilloscope PWM Pulse output in the PORTC pins when pressing the INT0, INT1, INT2 button to adjust the duty cycles

|  |  |  |
| --- | --- | --- |
| For Office use only | Signature of Lab Instructor | Remarks |
| Circuit Design in Proteus |  |  |
| Code Running in Proteus correctly |  |  |

II.3 TASK3 : LCD and EXTERNAL INTERUPT

II.3.1 Problem Statement: design a simple digital clock using Timer 1 Interrupt (Normal Mode and CTC mode) to count the seconds and develop a digital clock and display to LCD with the format hh:mm:ss. There are 3 buttons to adjust the digital clock. Button bt0 INT0 to adjust the hour digits, Button bt1 INT1 to adjust the minute digits, Button bt2 INT2 to adjust the second digits.

(Write the program in C code and Assembly) :



II.3.2 Circuit Design:

Student Design the application circuit in Proteus

II.3.3 Theory:

Student review related theory knowledge about timer1 and external interrupts

II.3.4 Algorithm

1. set up INT0 working with falling edge activation mode to adjust hour digits
2. set up INT1 working with falling edge activation mode to adjust minute digits
3. set up INT0 working with low level activation mode to to adjust second digits
4. Write main program

+ Setting up PORTC as output connected to LCD 4 bit mode

+ Enable all defined interrupts

+ While(1) loop:

Display digital clock in form hh:mm:ss

II.3.5 Code

|  |  |
| --- | --- |
| Write general comments about the program here  /\*----------------------------------------------------------------  C code  #define F\_CPU 8000000UL // XTAL = 8MHZ = 8000000Hz  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #define LCD\_DATA PORTC // port connected to LCD data pins  #define DATA\_DDR DDRC // direction register for data pins  #define LCD\_CTRL PORTC // port connected to LCD control pins  #define CTRL\_DDR DDRC // direction register for control pins  #define LCD\_RS 0 // define MCU pin connected to LCD RS  #define LCD\_RW 2 // define MCU pin connected to LCD R/W  #define LCD\_E 1 // define MCU pin connected to LCD E  #define LCD\_D0 0 // define MCU pin connected to LCD D0  #define LCD\_D1 1 // define MCU pin connected to LCD D1  #define LCD\_D2 2 // define MCU pin connected to LCD D1  #define LCD\_D3 3 // define MCU pin connected to LCD D2  #define LCD\_D4 4 // define MCU pin connected to LCD D3  #define LCD\_D5 5 // define MCU pin connected to LCD D4  #define LCD\_D6 6 // define MCU pin connected to LCD D5  #define LCD\_D7 7 // define MCU pin connected to LCD D6  void LCD\_Command(unsigned char cmd){ //Sends Command to LCD  //4 bit part  LCD\_DATA=(cmd&0b11110000); // send upper 4-bits  LCD\_CTRL|=1<<LCD\_E; // E=1 ,RS=0, RW=0  \_delay\_ms(1); // keep E=1 for some time  LCD\_CTRL&=~(1<<LCD\_E); // E=0 ,RS=0, RW=0  \_delay\_ms(1); // keep E=0 for some time  LCD\_DATA=((cmd&0b00001111)<<4); // send lover 4-bits  LCD\_CTRL|=1<<LCD\_E; // E=1,RS=0,RW=0  \_delay\_ms(1); // keep E=1 for some time  LCD\_CTRL&=~(1<<LCD\_E); // E=0,RS=0,RW=0  \_delay\_ms(1); // keep E=0 for some time  }  void delay1s(void){ //delay 1s  unsigned char i;  for(i=0;i<100;i++){  \_delay\_ms(10);  }  }  void LCD\_Show(uint8\_t ch)  { //Sends Char to LCD  LCD\_DATA=(ch&0b11110000); // send upper 4-bits  LCD\_CTRL|=(1<<LCD\_E)|(1<<LCD\_RS); // E=1, RS=1  \_delay\_ms(1); // keep E=1 for some time  LCD\_CTRL&=~((1<<LCD\_E)); // E=0  \_delay\_ms(1); // keep E=0 for some time    LCD\_DATA=((ch&0b00001111)<<4); // send lower 4-bits  LCD\_CTRL|= (1<<LCD\_E)|(1<<LCD\_RS); // E=1, RS=1  \_delay\_ms(1); // keep E=1 for some time  LCD\_CTRL&=~(1<<LCD\_E); // E=0  \_delay\_ms(1); // keep E=0 for some time  }  void LCD\_init(void){ //Initializes LCD  \_delay\_ms(15);  LCD\_DATA=0x00; // data = 0  LCD\_CTRL=0x00; // RS = RW = E = 0  DATA\_DDR|=1<<LCD\_D7|1<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4;  // Set bits 4 to 7 as output pins for data out  CTRL\_DDR|=1<<LCD\_E|1<<LCD\_RW|1<<LCD\_RS;  // Set bit 0 to 2 as output pins  //---------one------ // DATA = 0x30 ;  LCD\_DATA = 0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4; //4 bit mode  // E=1, RW=0, RS=0 for command mode  LCD\_CTRL|= 1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;  \_delay\_ms(1); // keep E=1 for some time  // E=0;  LCD\_CTRL&=~(1<<LCD\_E);  \_delay\_ms(1); // keep E=0 for some time  //-----------two----------- // DATA = 0x30 ;  LCD\_DATA=0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|1<<LCD\_D4;  //4 bit mode  // E=1, RW=0, RS=0 for command mode  LCD\_CTRL|=1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;  \_delay\_ms(1); // keep E=1 for some time  LCD\_CTRL&=~(1<<LCD\_E); // E=0  \_delay\_ms(1); // keep E=0 for some time  //-------three-------------  // DATA = 0x20 ;  LCD\_DATA=0<<LCD\_D7|0<<LCD\_D6|1<<LCD\_D5|0<<LCD\_D4; //4 bit mode  // E=1, RW=0, RS=0 for command mode  LCD\_CTRL|=1<<LCD\_E|0<<LCD\_RW|0<<LCD\_RS;  \_delay\_ms(1); // keep E=1 for some time  LCD\_CTRL&=~(1<<LCD\_E); // E=0  \_delay\_ms(1); // keep E=0 for some time  //--------4 bit--dual line---------------  LCD\_Command(0b00101000); // 0x28  //-----increment address, invisible cursor shift------  LCD\_Command(0b00001100); // 0x0C  LCD\_Command(0b10000000); // 0x80  }  void LCD\_String (char \*str)  {  int i;  for(i=0;str[i]!=0;i++) /\* send each char of string till the NULL \*/  {  LCD\_Show(str[i]); /\* call LCD data write \*/  }  }  void Time1\_initilazation()  {  TCNT1 = 34286;  TCCR1A = 0x00;  // timer1 in normal mode, prescaler/256  TCCR1B = 0x04; // use internal CLK.  TIMSK = (1<<TOIE1); // enable Timers 0 and 1 interrupts.  sei (); // enable global interrupts, set bit7 of SREG  }  unsigned char message1[] ="Didital Clock";  unsigned char message2[] ="00:00:00 ";  int hour=0,minute=0,second=0;  int main(void){  unsigned char i;  //Timer1 initilization  Time1\_initilazation();  MCUCR = 0x00; // make INT0 and INT1 low level triggered  MCUCSR = (1<<ISC2); // make INT2 rising edge triggered  GICR = (1<<INT0)|(1<<INT1)|(1<<INT2);  sei (); // enable interrupts  LCD\_init();//init LCD bit, dual line, cursor right  LCD\_Command(0x01); //clears LCD, Cursor at Home  delay1s();  LCD\_Command(0x80); // cursor at upper line most left  LCD\_String (message1) ;  while(1)  { //loop for ever  sprintf(message2, "%0.2d:%0.2d:%0.2d",hour,minute,second);  LCD\_Command(0xC0); // cursor at lower line most left  LCD\_String (message2);  }  return 0;  }  ISR (TIMER1\_OVF\_vect){ // ISR for Timer1 overflow  TCNT1 =34286;  second++;  if(second==60)  {  second=0;  minute++;  if(minute==60)  {  minute=0;  hour++;  if(hour==24)  hour=0;  }  }    }  ISR (INT0\_vect){ // ISR for external interrupt 0  second++;  if(second == 60) { // Check if the hour reaches 24  second = 0; // Reset the hour to 0  }  }  ISR (INT1\_vect){ // ISR for external interrupt 1  minute++;  if(minute == 60) { // Check if the hour reaches 24  minute = 0; // Reset the hour to 0  }  }  ISR (INT2\_vect){ // ISR for external interrupt 2  hour++;  if(hour == 24) { // Check if the hour reaches 24  hour = 0; // Reset the hour to 0  }  }  This program is a multi-tasks program to control a simple Alarm clock  Author: xxxxxxxxx Date: xx.xx.20xx  ---------------------------------------------------------------------\*/ | |
| Instruction | Comments |
|  |  |

II.3.6 Result

Check the working of digital clock circuit.

|  |  |  |
| --- | --- | --- |
| For Office use only | Signature of Lab Instructor | Remarks |
| Circuit Design in Proteus |  |  |
| Code Running in Proteus correctly |  |  |

II.4 TASK4 : TIMERS, LCD and EXTERNAL INTERUPT

II.4.1 Problem Statement:

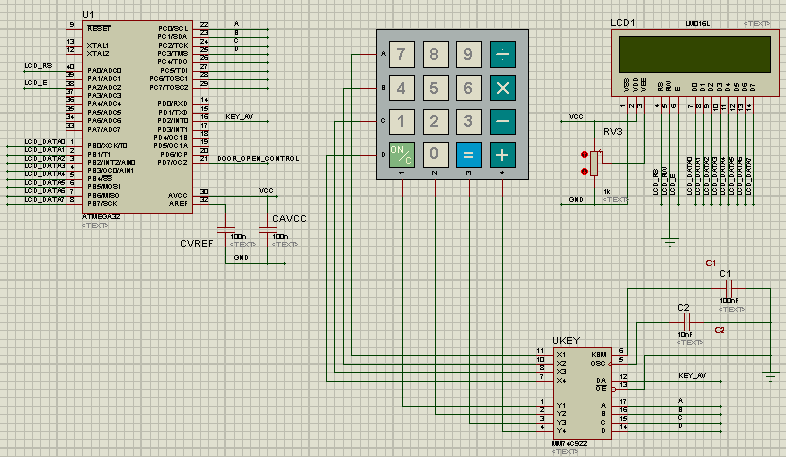
Write An AVR Assembler Program to control a door lock system (with given circuit design) which have following Requirements:

- LCD to display Data

- Keypad to input password

- Button for changing to Password Change Mode and Normal Mode

- Password must be stored in EEPROM



II.4.2 Theory:

Student review related theory knowledge about GPIO and external interrupts

II.4.3 Algorithm

1. Initilze CLD
2. set up INT0 working with falling edge activation mode to read keypad input
3. Set up PORTC as input
4. set up Door Lock Control as ouput pin
5. Write main program

+ Setting up PORTC as output connected to LCD 4 bit mode

+ Enable defined interrupts

+ While(1) loop:

Display password data

ISR(INT0)

* Read data from keypad
* Process input keypad

II.4.3 Code

|  |  |
| --- | --- |
| Write general comments about the program here  /\*----------------------------------------------------------------  This program is …..  Code C  #define F\_CPU 1000000UL // XTAL = 16MHZ = 8000000Hz  #include <avr/io.h>  #include <util/delay.h>  #include <avr/interrupt.h>  #define RS 0 // bit 5 of Port  #define RW 1 // bit 6 of Port  #define E 2 // bit 7 of Port  #define DATA\_BUS PORTB  #define DATA\_DDR DDRB  #define CTRL\_BUS PORTA  #define CTRL\_DDR DDRA  void delay(unsigned int z) // For delay when LCD Starts  { unsigned int x;  for(x=0 ; x<z ; x++)  \_delay\_ms(10);  } //Delay Function Ends  int ready(){ //For checking that the LCD is ready or not?  delay(10); return 1;  } //Ready Function Ends  void LCD\_Pulse\_E(int t){  CTRL\_BUS |= 0b00000100 ; // E = 1;  delay(t);  CTRL\_BUS &= 0b11111011 ; // E = 0;  delay(t);  }  int LCD\_Command(unsigned char COMMAND){  ready();  DATA\_BUS = COMMAND;  CTRL\_BUS = 0b11111000;  LCD\_Pulse\_E(1);  return 1;  }  int LCD\_Show(unsigned char CHARACTER){  ready();  DATA\_BUS = CHARACTER;  CTRL\_BUS = 0b11111001 ;  // RS=1; RW=0;  LCD\_Pulse\_E(1);  return 1;  }  void LCD\_String (char \*str)  {  int i;  for(i=0;str[i]!=0;i++) /\* send each char of string till the NULL \*/  {  LCD\_Show(str[i]); /\* call LCD data write \*/  }  }  void N\_char\_LCD\_String (char \*str,int pos, int n)  {  int i;  for(i=pos;i<n+pos;i++) /\* send each char of string till the NULL \*/  {  LCD\_Show(str[i]); /\* call LCD data write \*/  }  }  int LCD\_Initialize(){  LCD\_Command(0x38); // 8 data lines, two lines, Font 5x7.  LCD\_Command(0x0F); // Display=ON, Curson=ON, Cursor Blonking=ON  LCD\_Command(0x01); // Clear display and return cursor to the home position  LCD\_Command(0x06); // During read/write operation only cursor (not text)  // should move right.  LCD\_Command(0x80); // Cursor at Line 1, Position 0  return 1;  }  unsigned char data1[]="Door Lock";  unsigned char data2[]="Study at School of Computer Engineering " ;  save\_password[]="12345";  input\_password[]=" ";  pass\_index=0;  int main()  {    DATA\_DDR = 0xFF;  CTRL\_DDR = 0xFF;  CTRL\_BUS = 0;  DATA\_BUS = 0;  DDRD=0x00; / configure PORTD input    DDRC = 0x00; // configure PORTC input  MCUCR = 0b00000010; // make INT0 falling triggered  GICR = (1<<INT0);  sei ();  delay(500); // wait for LCD to Start  LCD\_Initialize();  LCD\_Command(0x04); // Clear display and return cursor to line 1,col 1 position  LCD\_String(data1);  LCD\_Command(0xC0); // Clear display and return cursor to line 2,col 1 position  while(1);  return 0;  }  ISR (INT0\_vect){ // ISR for external interrupt 0  char key\_input,decode\_key=0;  key\_input=PINC&0x0F;  switch(key\_input)  {  case 0: decode\_key='7';  break;  case 1: decode\_key='4';  break;  case 2: decode\_key='1';  break;  case 3: decode\_key='C';  break;  case 4: decode\_key='8';  break;  case 5: decode\_key='5';  break;  case 6: decode\_key='2';  break;  case 7: decode\_key='0';  break;  case 8: decode\_key='9';  break;  case 9: decode\_key='6';  break;  case 10: decode\_key='3';  break;  case 11: decode\_key='=';  break;  case 12: decode\_key='/';  break;  case 13: decode\_key='x';  break;  case 14: decode\_key='-';  break;  case 15: decode\_key='+';  break;  }  input\_password[pass\_index]=decode\_key;  pass\_index++;  LCD\_Show(decode\_key);  if(pass\_index==5)  {  if(strncmp(input\_password, save\_password,5)==0)  {  PORTD |=(1<<7);  LCD\_Command(0x04); // Clear display and return cursor to line 1,col 1 position  LCD\_String("correct Pass");  }  else  {  PORTD &=(~(1<<7));  PORTD |=(1<<7);  LCD\_Command(0x04); // Clear display and return cursor to line 1,col 1 position  LCD\_String("invalid Pass");  }  pass\_index=0;  }  }  Author: xxxxxxxxx Date: xx.xx.20xx  ---------------------------------------------------------------------\*/ | |
| Instruction | Comments |
|  |  |

II.4.5 Result

|  |  |  |
| --- | --- | --- |
| For Office use only | Signature of Lab Instructor | Remarks |
| Circuit Design in Proteus |  |  |
| Code Running in Proteus correctly |  |  |

III. REPPORT

After finish all tasks student analyze the laboratory results and submit your report in blackboard. The report for each task should include

* Theory ( Related theory)
* Circuit diagram
* Algorithm
* Code with clear comments

The format of the report should follow the Lab report guideline of School of Electrical Engineering

**GRADING GUIDELINE FOR LAB REPORT**

Name of Student: ID:

Subject: Lab Number:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number** | **Content** |  | **Score** | **Comment** |
| 1 | **Format (max 9%)** | |  |  |
| * Font type | Yes No |  |
| * Font size | Yes No |  |
| * Lab title | Yes No |  |
| * Page number | Yes No |  |
| * Table of contents | Yes No |  |
| * Header/Footer | Yes No |  |
| * List of figures (if exists) | Yes No |  |
| * List of tables (if exists) | Yes No |  |
| * Lab report structure | Yes No |  |
| 2 | **English Grammar and Spelling (max 6%)** | |  |  |
| * Grammar | Yes No |  |
| * Spelling | Yes No |  |
| 3 | **Data and Result Analysis (max 85%)**   * Running result check * Report contents * Code with clear comments | |  |  |
| **Total Score** | |  | |  |

Signature:

Date: