Embedded Lock System Documentation

Table of Contents

Overview

The Embedded Lock System is a collaborative project designed to implement a secure lock system. The system utilizes Proteus 8 Professional for simulation and CodeVisionAVR Evaluation for programming the ATmega16 microcontroller. Written in the C programming language, the system encompasses three main functionalities: opening the door, setting a new passcode (PC), and accessing administrative features. The project is organized into three distinct parts, with each part expertly handled by different contributors.

Features

- Password-based Access Control
- LCD Display for User Interaction
- Audible Alarms for Incorrect Entries

Define interrupts priorities:

(Recommended):

- Press the Open button to open the door, triggering button '*'.
- Prioritize the Admin button by associating it with interrupt INT0.

- Set the PC configuration with the Set PC button, utilizing interrupt INT1.

> The project favors this Option because it assigns higher priority to the Admin button, followed logically by the > Set PC button, and then the Open button.

Getting Started

Prerequisites

Ensure you have the following tools and components:

- Proteus 8 Professional
- CodeVisionAVR Evaluation
- ATmega16 Microcontroller
- Other necessary components (LCD, DC Motor, Buzzer, Keypad)

Installation

1. Clone the repository:

```
git clone https://github.com/Hussein119/lock-system.git
cd lock-system
```

2. Open the project in CodeVisionAVR.
    - Launch CodeVisionAVR and open the project file (`\Code\Project #1 lock system.prj`).
    - Customize project settings if necessary.
3. Simulate in Proteus.
    - Open Proteus 8 Professional.
    - Load the simulation file (`\Simulation\Project #1 lock system.pdsprj`) and run the simulation.
4. Hardware Implementation.
    - Connect the ATmega16 to the necessary components.
    - Program the microcontroller using CodeVisionAVR.

Usage

Test the lock system with the predefined password, verify LED indicators, and explore other functionalities.

Proteus Simulation

Hardware Components

1. ATmega16 Microcontroller
2. LCD Display
3. Keypad 4x3
4. Red light (for door simulation)
5. Speaker (Peeps alarm)
6. Keypad 4x3
7. Three Buttons for interrupts

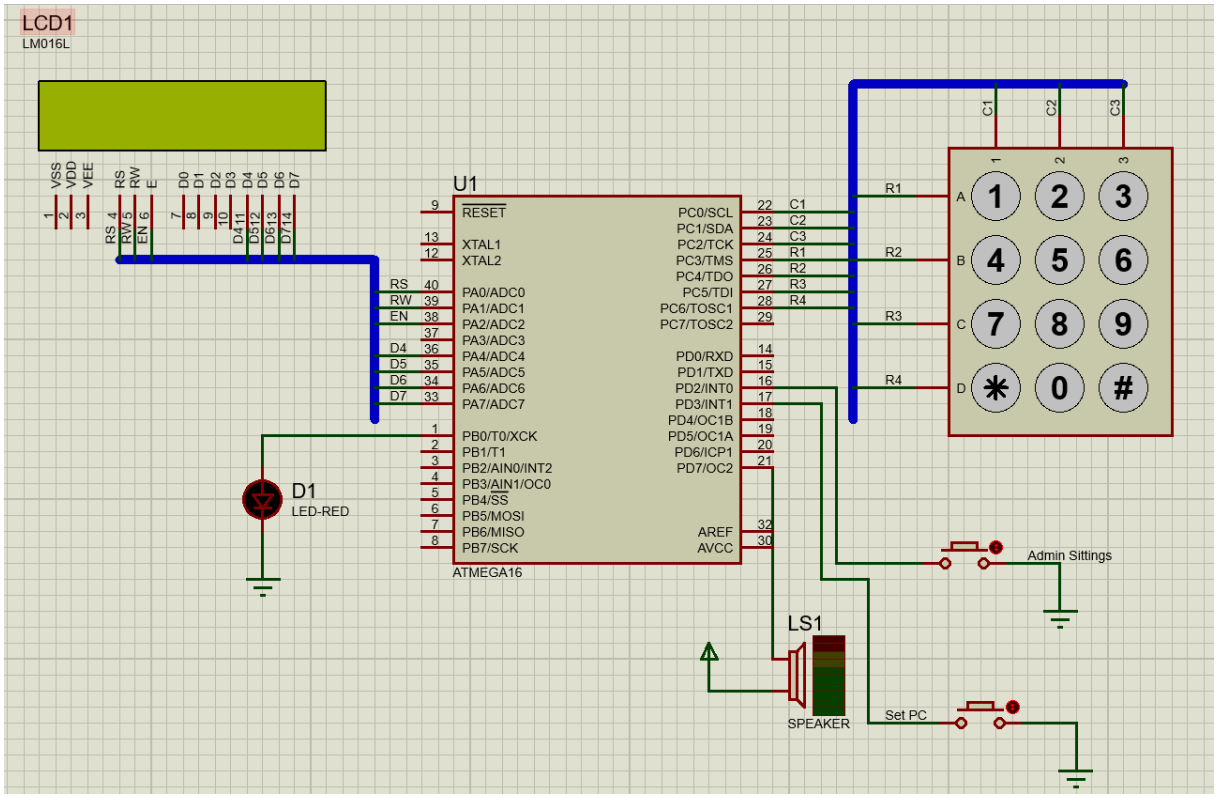

Figure 1: Hardware

Main Program Flowchart

Open Door Flowchart

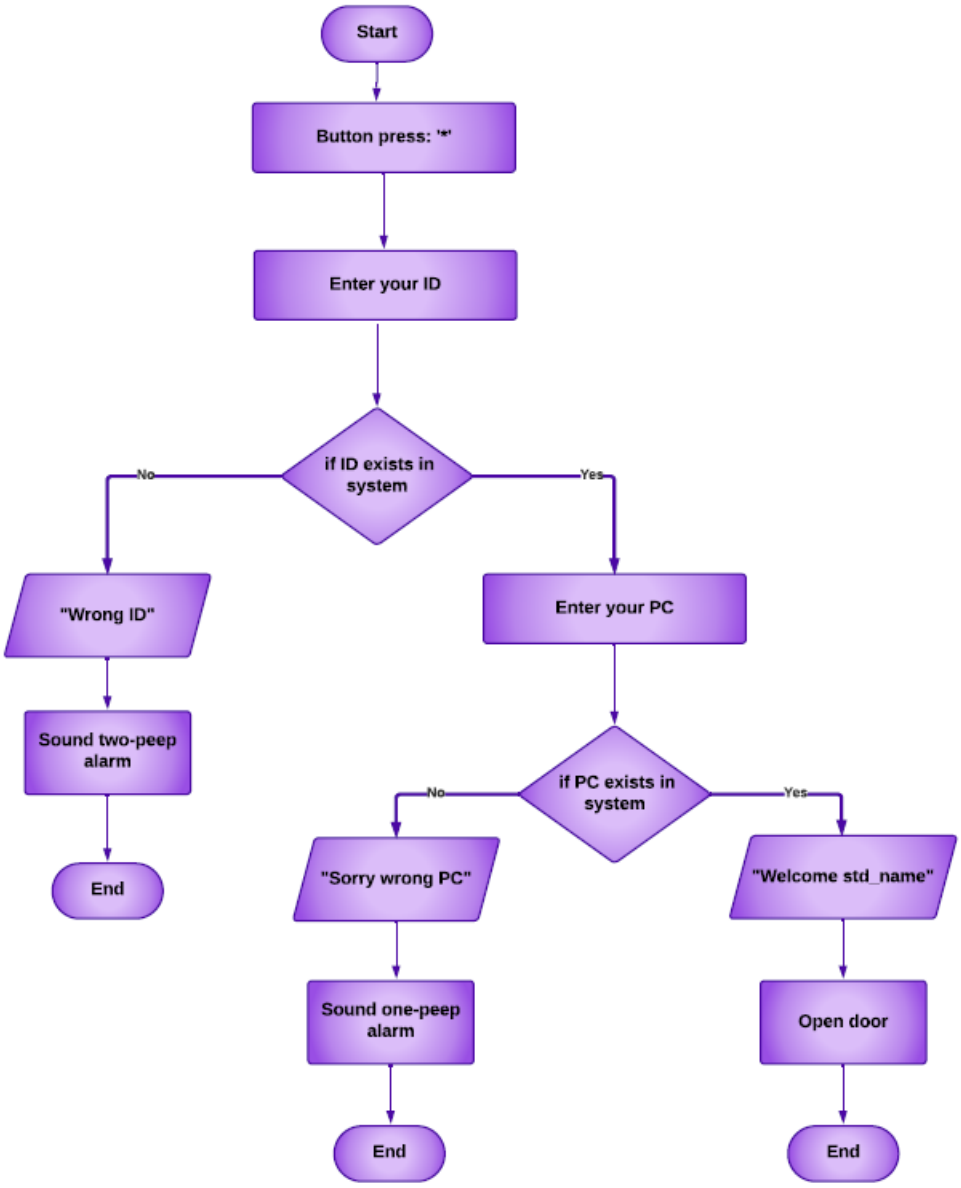

Figure 2: Open Door

Set New PC Flowchart

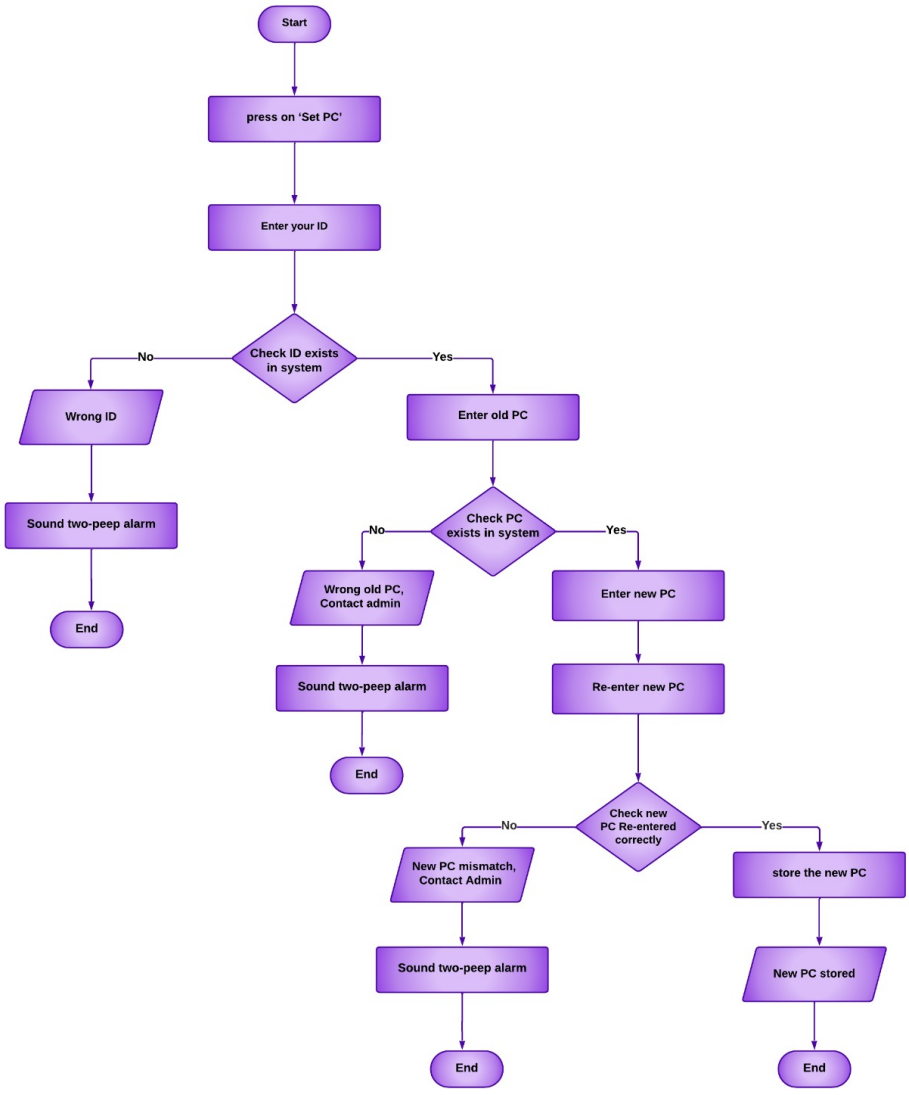

Figure 3: Set New PC

Admin Sittings Flowchart

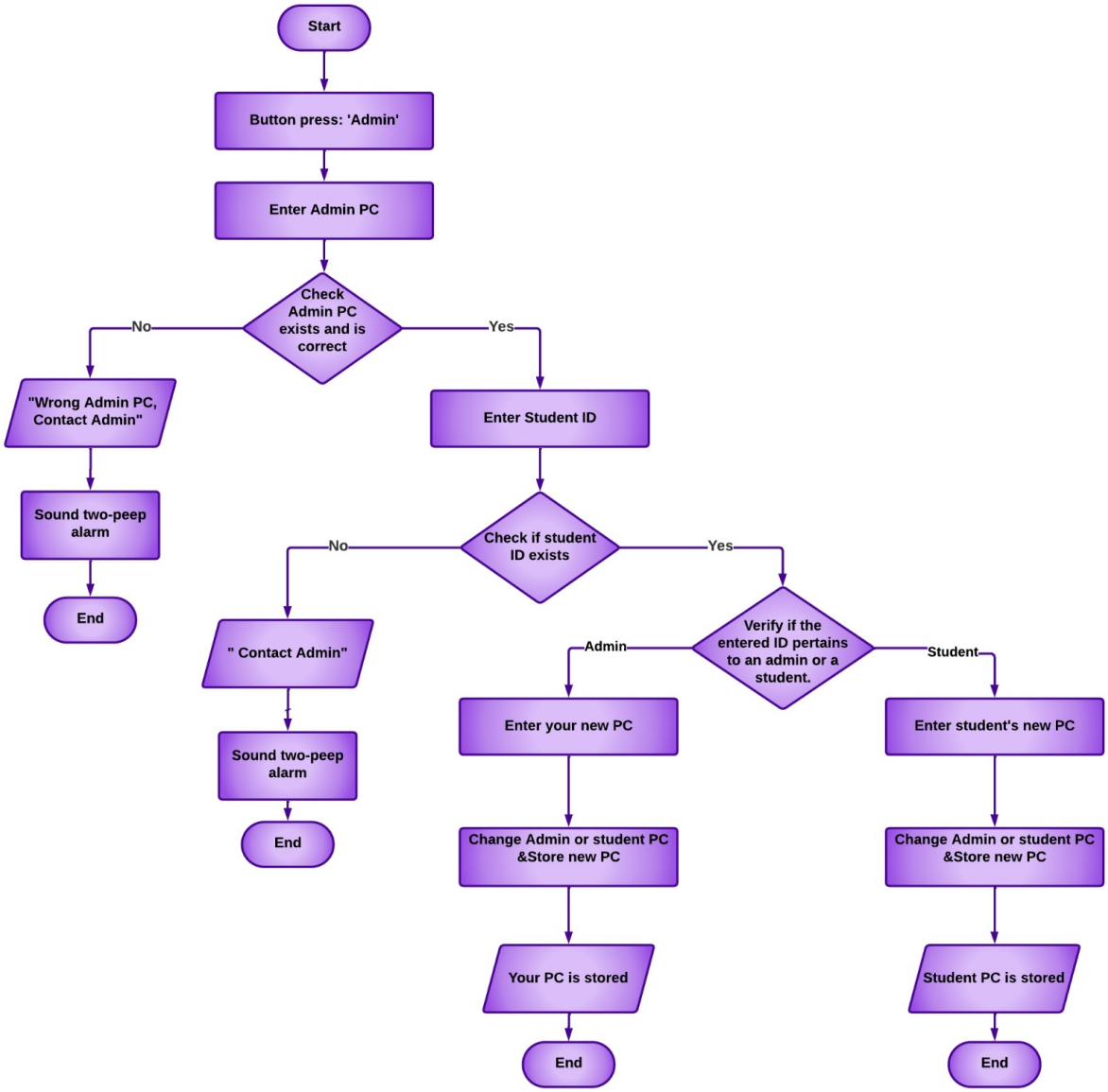

Figure 4: Admin Sittings

Developers

- Islam AbdElhady Hassanein
- Ahmed Hesham Fathall Farag
- Elsherif Shapan Abdelmageed

- Hussein AbdElkader Hussein
- Enas Ragab AbdEllatif
- Mariam Tarek Saad

Main Function

- LockSystem.c

```c
/*
 * Project #1 lock system.c
 *
 * Created: 12/16/2023 1:47:34 AM
 * Author: Hos10
 */

#include "lockSys.h"

void main(void)
{
    char input;

    // Initialize Hardware
    initializeHardware();

    // Initialize user data in EEPROM
    initializeUsers();

    // Initialize interrupts for various modes
    initializeIntrrupts();

    // If user need to open the door must press '*' on the keypad
    while (1)
        {
        input = keypad();
        if (input != '*')
            continue;
        openCloseDoorMode();
        }
}

interrupt[3] void setPC(void) //  vector no 3 -> INT1
{
    setPCMode();
}

interrupt[2] void admin(void) // vector no 2 -> INT0
{
    adminMode();
}
```

Header File

- lockSys.h

```c
#include <mega16.h>
#include <alcd.h>
#include <delay.h>
#include <string.h>

// Macros for setting and clearing bits in a register
#define bit_set(r, b) r |= 1 << b
#define bit_clr(r, b) r &= ~(1 << b)

// Function prototypes
void initializeHardware();
void initializeKeypad();
void initializeDoor();
void initializeSpeaker();
void initializeIntrrupts();
char keypad();
unsigned char EE_Read(unsigned int address);
void EE_Write(unsigned int address, unsigned char data);
void EE_WriteString(unsigned int address, const char *str);
void EE_ReadString(unsigned int address, char *buffer, unsigned int
    length);
void initializeUsers();
void displayMessage(char *message, int delay_ms_value);
int enterValueWithKeypad(char *buffer);
void generateTone();
void adminMode();
void setPCMode();
void openCloseDoorMode();

// User structure to store user data
typedef struct
{
    char name[6];
    char id[4];
    char pc[4];
} User;
// Array of user data
User users[] =
{
        // name   ID     PC
        {"Prof", "111", "203"},
        {"Ahmed", "126", "129"},
        {"Amr", "128", "325"},
```

```
        {"Adel", "130", "426"},
        {"Omer", "132", "079"},
};

// Function to initialize hardware components
void initializeHardware()
{
    initializeKeypad();
    lcd_init(16); // Initialize the LCD
    initializeDoor();
    initializeSpeaker();
}

// Function to initialize keypad
void initializeKeypad()
{
    // Set keypad ports
    DDRC = 0b00000111;  // 1 unused pin, 4 rows (input), 3 columns (
        output)
    PORTC = 0b11111000; // pull-up resistance
}

// Function to initialize door
void initializeDoor()
{
    // Set the door as input (by default, the door is closed)
    DDRB .0 = 0;
    PORTB .0 = 1; // turn on pull-up resistance
}

// Function to initialize speaker
void initializeSpeaker()
{
    // Set the speaker as an output
    DDRD .7 = 1;
    PORTD .7 = 1; // Set it to 1 initially
}

// Function to initialize interrupts
void initializeIntrrupts()
{
    DDRB .2 = 0;  // make button as input
    PORTB .2 = 1; // turn on pull up resistance for INT2 intrrupt

    // actual casue INT2
    bit_set(MCUCSR, 6);

    DDRD .2 = 0;  // make button as input
    PORTD .2 = 1; // turn on pull up resistance for INT0 intrrupt

    // actual casue (The falling edge of INT0)
```

```
93      bit_set(MCUCR, 1);
94      bit_clr(MCUCR, 0);
95
96      // actual casue (The falling edge of INT1)
97      bit_set(MCUCR, 3);
98      bit_clr(MCUCR, 2);
99
100     DDRD .3 = 0;  // make button SetPC as input
101     PORTD .3 = 1; // turn on pull up resistance
102
103     // Enable global interrupts
104 #asm("sei")
105
106     // GICR INT1 (bit no 7) , SetPC spacific enable
107     bit_set(GICR, 7);
108
109     // GICR INT2 (bit no 5) , open spacific enable
110     bit_set(GICR, 5);
111
112     // GICR INT0 (bit no 6) , admin spacific enable
113     bit_set(GICR, 6);
114 }
115
116 // Function: keypad
117 // Description: Reads the input from a 4x3 matrix keypad and returns
        the corresponding key value.
118 //              The keypad is connected to port C, and the function
        scans each row and column
119 //              combination to determine the pressed key.
120 // Returns: Character representing the pressed key.
121 char keypad()
122 {
123     while (1)
124     {
125         PORTC .0 = 0;
126         PORTC .1 = 1;
127         PORTC .2 = 1;
128
129         switch (PINC)
130         {
131         case 0b11110110:
132             while (PINC .3 == 0)
133                 ;
134             return 1;
135         case 0b11101110:
136             while (PINC .4 == 0)
137                 ;
138             return 4;
139         case 0b11011110:
140             while (PINC .5 == 0)
141                 ;
```

```
142             return 7;
143         case 0b10111110:
144             while (PINC .6 == 0)
145                 ;
146             return '*';
147         }
148 
149         PORTC .0 = 1;
150         PORTC .1 = 0;
151         PORTC .2 = 1;
152 
153         switch (PINC)
154         {
155         case 0b11110101:
156             while (PINC .3 == 0)
157                 ;
158             return 2;
159         case 0b11101101:
160             while (PINC .4 == 0)
161                 ;
162             return 5;
163         case 0b11011101:
164             while (PINC .5 == 0)
165                 ;
166             return 8;
167         case 0b10111101:
168             while (PINC .6 == 0)
169                 ;
170             return 0;
171         }
172 
173         PORTC .0 = 1;
174         PORTC .1 = 1;
175         PORTC .2 = 0;
176 
177         switch (PINC)
178         {
179         case 0b11110011:
180             while (PINC .3 == 0)
181                 ;
182             return 3;
183         case 0b11101011:
184             while (PINC .4 == 0)
185                 ;
186             return 6;
187         case 0b11011011:
188             while (PINC .5 == 0)
189                 ;
190             return 9;
191         case 0b10111011:
192             while (PINC .6 == 0)
```

```
                ;
            return 11;
        }
    }
}

// Function to read from EEPROM
unsigned char EE_Read(unsigned int address)
{
    while (EECR .1 == 1); // Wait till EEPROM is ready
    EEAR = address; // Prepare the address you want to read from
    EECR .0 = 1;    // Execute read command
    return EEDR;
}

// Function to write to EEPROM
void EE_Write(unsigned int address, unsigned char data)
{
    while (EECR .1 == 1); // Wait till EEPROM is ready
    EEAR = address; // Prepare the address you want to read from
    EEDR = data;    // Prepare the data you want to write in the
       address above
    EECR .2 = 1;    // Master write enable
    EECR .1 = 1;    // Write Enable
}

// Function to write a string to EEPROM
void EE_WriteString(unsigned int address, const char *str)
{
    // Write each character of the string to EEPROM
    while (*str)
    EE_Write(address++, *str++);
    // Terminate the string with a null character
    EE_Write(address, '\0');
}

// Function to read a string from EEPROM
void EE_ReadString(unsigned int address, char *buffer, unsigned int
   length)
{
    unsigned int i;
    for (i = 0; i < length; ++i)
    {
        buffer[i] = EE_Read(address + i);
        if (buffer[i] == '\0')
            break;
    }
}

// Function to initialize user data in EEPROM
void initializeUsers()
```

```
{
    unsigned int address = 0;
    int i;
    for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
    {
        EE_WriteString(address, users[i].name);
        address += sizeof(users[i].name);

        EE_WriteString(address, users[i].id);
        address += sizeof(users[i].id);

        EE_WriteString(address, users[i].pc);
        address += sizeof(users[i].pc);
    }
}

// Function to display a message on the LCD
void displayMessage(char *message, int delay_ms_value)
{
    lcd_clear();
    lcd_puts(message);
    delay_ms(delay_ms_value);
}

// Function to enter a value with the keypad
int enterValueWithKeypad(char *buffer)
{
    buffer[0] = keypad() + '0';
    lcd_putchar(buffer[0]);
    buffer[1] = keypad() + '0';
    lcd_putchar(buffer[1]);
    buffer[2] = keypad() + '0';
    lcd_putchar(buffer[2]);
    buffer[3] = '\0'; // Null-terminate the string

    delay_ms(1000);

    return 1; // Return a non-zero value to indicate success
}

// Function to generate a tone with speaker
void generateTone()
{
    PORTD .7 = 1;
    delay_ms(500);
    PORTD .7 = 0;
    delay_ms(500);
    PORTD .7 = 1;
}

// Interrupt functions
```

```
293
294 // Function for admin mode
295 void adminMode()
296 {
297     char enteredPC[4];
298     char enteredStudentID[4];
299     char enteredNewPC[4];
300     User student;
301     User admin;
302     unsigned int adminPCAddress = 0;
303     unsigned int address = 0;
304     int userFound = 0;
305     int i;
306
307     for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
308     {
309         EE_ReadString(address, admin.name, sizeof(users[i].name));
310         if (strcmp(admin.name, "Prof") == 0)
311         {
312             address += sizeof(users[i].name);
313             EE_ReadString(address, admin.id, sizeof(admin.id));
314             address += sizeof(users[i].id);
315             EE_ReadString(address, admin.pc, sizeof(admin.pc));
316             adminPCAddress = address;
317             break;
318         }
319         address += sizeof(users[i].pc);
320     }
321
322     address = 0; // reset the address
323
324     displayMessage("Enter Admin PC: ", 1000);
325     lcd_gotoxy(0, 1);
326
327     if (enterValueWithKeypad(enteredPC))
328     {
329
330         if (strcmp(admin.pc, enteredPC) == 0)
331         {
332             displayMessage("Enter Student ID: ", 1000);
333
334             if (enterValueWithKeypad(enteredStudentID))
335             {
336                 int j;
337                 for (j = 0; j < sizeof(users) / sizeof(users[0]); ++j)
338                 {
339                     address += sizeof(users[j].name);
340                     EE_ReadString(address, student.id, sizeof(student.
                        id));
341                     address += sizeof(users[j].id);
342                     if (strcmp(student.id, enteredStudentID) == 0)
```

```
343                         {
344                             displayMessage("Enter student's new PC: ",
                                    1000);
345                             if (enterValueWithKeypad(enteredNewPC))
346                             {
347                                 // Set the new pc for this student, address
                                        is for student PC
348                                 EE_WriteString(address, enteredNewPC);
349                                 displayMessage("Student PC is stored",
                                    3000);
350                                 userFound = 1;
351                                 break;
352                             }
353                         }
354                         else if (strcmp(admin.id, enteredStudentID) == 0)
355                         {
356                             displayMessage("Enter your new PC: ", 1000);
357                             lcd_gotoxy(0, 1);
358                             if (enterValueWithKeypad(enteredNewPC))
359                             {
360                                 // Set the new pc for this user (Admin),
                                    address is for admin PC
361                                 EE_WriteString(adminPCAddress, enteredNewPC
                                    );
362                                 displayMessage("Your PC is stored", 3000);
363                                 userFound = 1;
364                                 break;
365                             }
366                         }
367                         address += sizeof(users[i].pc);
368                     }
369                 }
370             }
371         }
372 
373         if (!userFound)
374         {
375             displayMessage("Contact Admin", 3000);
376             // Two peeps alarm
377             generateTone();
378             generateTone();
379         }
380         delay_ms(5000);
381         lcd_clear();
382     }
383 
384     // Function for set PC mode
385     void setPCMode()
386     {
387         char enteredID[5]; // Change data type to string
388         User currentUser;
```

```
389     unsigned int address = 0;
390     int userFound = 0;
391     int i;
392     char enteredNewPC[5];   // define enteredNewPC array to hold the
            new PC
393     char reenteredNewPC[5]; // define reenteredNewPC array to hold the
            Re-entered new PC
394 
395     lcd_clear();
396     displayMessage("Enter your ID:", 1000);
397     lcd_gotoxy(0, 1);
398     if (enterValueWithKeypad(enteredID))
399     {
400         char enteredOldPC[5];
401         // search for the entered ID in the user data
402         for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
403         {
404             address += sizeof(users[i].name);
405             EE_ReadString(address, currentUser.id, sizeof(currentUser.
                id)); // Read ID as a string
406 
407             if (strcmp(currentUser.id, enteredID) == 0)
408             {
409                 // ID found, verify the old PC
410                 address += sizeof(currentUser.id);
411                 EE_ReadString(address, currentUser.pc, sizeof(
                    currentUser.pc)); // Read PC as a string
412                 displayMessage("Enter old PC:", 1000);
413                 lcd_gotoxy(0, 1);
414 
415                 if (enterValueWithKeypad(enteredOldPC))
416                 {
417                     if (strcmp(currentUser.pc, enteredOldPC) == 0)
418                     {
419                         // Old PC verified
420                         displayMessage("Enter new PC:", 1000);
421                         lcd_gotoxy(0, 1);
422                         enterValueWithKeypad(enteredNewPC);
423 
424                         lcd_clear();
425                         displayMessage("Re-enter new PC:", 1000);
426                         lcd_gotoxy(0, 1);
427                         enterValueWithKeypad(reenteredNewPC);
428 
429                         if (strcmp(enteredNewPC, reenteredNewPC) == 0)
430                         {
431                             // If new PC entered correctly, store it
432                             EE_WriteString(address, enteredNewPC);
433                             displayMessage("New PC stored", 1000);
434                         }
435                         else
```

```
436                             {
437                                 displayMessage("New PC mismatch, Contact
                                        admin", 1000);
438                                 generateTone();
439                                 generateTone();
440                             }
441                         }
442                         else
443                         {
444                             displayMessage("Wrong old PC,   Contact admin",
                                    1000);
445 
446                             generateTone();
447                             generateTone();
448                         }
449                     }
450 
451                     userFound = 1;
452                     break;
453                 }
454 
455                 address += sizeof(users[i].id);
456                 address += sizeof(users[i].pc);
457             }
458 
459             if (!userFound)
460             {
461                 displayMessage("Wrong ID", 1000);
462                 generateTone();
463                 generateTone();
464             }
465             delay_ms(5000);
466             lcd_clear();
467         }
468 }
469 
470 // Function for open/close door mode
471 void openCloseDoorMode()
472 {
473     char enteredID[4]; // Change data type to string
474     User currentUser;
475     unsigned int address = 0;
476     int userFound = 0;
477     int i;
478 
479     displayMessage("Enter your ID: ", 1000);
480     lcd_gotoxy(0, 1);
481 
482     if (enterValueWithKeypad(enteredID))
483     {
484         char enteredPC[4];
```

```
485         for (i = 0; i < sizeof(users) / sizeof(users[0]); ++i)
486         {
487             EE_ReadString(address, currentUser.name, sizeof(users[i].
                name));
488             address += sizeof(users[i].name);
489             EE_ReadString(address, currentUser.id, sizeof(currentUser.
                id)); // Read ID as a string
490 
491             if (strcmp(currentUser.id, enteredID) == 0)
492             {
493 
494                 address += sizeof(users[i].id);
495                 EE_ReadString(address, currentUser.pc, sizeof(
                    currentUser.pc)); // Read PC as a string
496 
497                 displayMessage("Enter your PC: ", 1000);
498                 lcd_gotoxy(0, 1);
499 
500                 if (enterValueWithKeypad(enteredPC))
501                 {
502                     if (strcmp(currentUser.pc, enteredPC) == 0)
503                     {
504                         lcd_clear();
505                         lcd_puts("Welcome, ");
506                         lcd_puts(currentUser.name);
507                         // Open the door
508                         DDRB .0 = 1;
509                     }
510                     else
511                     {
512                         displayMessage("Sorry wrong PC", 1000);
513                         // one peep alarm
514                         generateTone();
515                     }
516                 }
517                 userFound = 1;
518                 break;
519             }
520 
521             address += sizeof(users[i].id);
522             address += sizeof(users[i].pc);
523         }
524     }
525 
526     if (!userFound)
527     {
528         displayMessage("Wrong ID", 1000);
529         // Two peeps alarm
530         generateTone();
531         generateTone();
532     }
```

```
533       delay_ms(5000);
534       // close the door and clear lcd
535       DDRB .0 = 0;
536       lcd_clear();
537   }
```