Palestine Polytechnic University

College of Engineering and Technology

Electrical and Computer Engineering Department

Graduation Project

Phone Directory System (PDS)

Project Team

Amjad Abu-Hassan

Mohammad Aqabneh

Murad Awawdeh

Project Supervisor

Eng. Khalid Daghameen

Hebron – Palestine

2010 – 2011

جامعة بوليتكنك فلسطين

الخليل – فلسطين

كلية الهندسة و التكنولوجيا

دائرة الهندسة الكهربائية و الحاسوب

اسم المشروع:

Phone Directory System (PDS)

أسماء الطلبة:

أمجد ماجد أبو حسان محمد إبراهيم عقابنة مراد محمود عواوده

بناء على نظام كلية الهندسة و التكنولوجيا و إشراف و متابعة المشرف المباشر على المشروع وموافقة أعضاء اللجنة الممتحنة تم تقديم هذا المشروع إلى دائرة الهندسة الكهربائية و الحاسوب وذلك للوفاء بمتطلبات درجة البكالوريوس في الهندسة تخصص هندسة أنظمة الحاسوب.

توقيع المشرف

.....

توقيع اللجنة الممتحنة

....

توقيع رئيس الدائرة

.....

Abstract

The Phone Directory System (PDS) is a system that its main function is to search for a phone number. It mainly consists of two parts. The first part is a remote server that holds the phone directory database. The second part is a Phone Directory device(PDD).

PDD provides a good user interface as it has a keypad to let the user enter the name of person to search for his number and it has a screen to view results. The results is returned and viewed on a screen. PDD is able to connect to the remote server to perform the search process in that server.

الإهـــداء

إلى من جرع الكأس فارغاً ليسقيني قطرة حب

إلى من كلّت أنامله ليقدم لنا لحظة سعادة

إلى من حصد الأشواك عن دربي ليمهد لي طريق العلم

إلى القلب الكبير **(والدي العزيز)**

إلى من أرضعتني الحب والحنان

إلى رمز الحب وبلسم الشفاء

إلى القلب الناصع بالبياض **(والدتي الحبيبة)**

إلى القلوب الطاهرة الرقيقة والنفوس البريئة إلى رياحين حياتي **(إخوتي)**

إلى الروح التي سكنت روحي

الآن تفتح الأشرعة وترفع المرساة لتنطلق السفينة في عرض بحر واسع مظلم هو بحر الحياة وفي هذه الظلمة لا يضيء إلا قنديل الذكريات

ذكريات الأخوة البعيدة إلى الذين أحببتهم وأحبوني **(أصدقائي)**

شكر وتقدير

الشكر أولا لله عز وجل

نتقدم بجزيل الشكر والامتنان إلى الصرح العلمي جامعة بوليتكنك فلسطين وإلى الهيئة التدريسية في دائرة الهندسة الكهربائية والحاسوب ونخص بالذكر مشرف المشروع:

أ. خالد الدغامين

كما نتقدم بالشكر إلى كل من:

أ. مازن زلوم

أ. أيمن وزوز

م. سامي السلامين

وإلى كل من مد يد العون لإنجاز هذا العمل .

Contents

Chapter Two : Theoretical Background

Chapter Three : Design Concepts

Chapter Four : Detailed Technical Project Design

Chapter Five : PDD Software

Chapter Six : System Implementation and Testing

List of Figures

List of Tables

Chapter One

Introduction

Chapter One

Introduction

This chapter gives an overview about the project, its importance, objectives, time plan, estimated cost and risk management.

1.1 Overview

The Phone Directory System(PDS) is a system that is attached to the phone. It has a phone directory device(PDD) that is connected with internet. Its main function is to search for a phone number for a specific name. PDD provides a good user interface since it has a keypad to enter a name and it has a LCD screen to view results.

The user has to enter the name that he is looking for his phone number using the keypad. Then start the search process. After that , the system has to connect to the internet. Then it sends a request to a specific remote server. The remote server holds a database containing names ,addresses and phone numbers . Then the system make a query in the server and return the results and view them on the LCD screen . If there is more than one result , the user has to select one .

1.2 Project Importance

The most important issue of this project is to provide easiness ,comfort and quickness for the telephone users. The following scenarios shows that. Suppose that a telephone user wants to call someone:

Scenario one : suppose that the user has a phone directory book. Each phone directory book has a certain methodology of search; some of them may be based on family name, others may be based on region. The user will start turning papers. He may not find the number because the phone book is not up to date or he may give up or change his mind or he may decide to use another scenario.

Scenario two : suppose that the user will call the customer service of the telecommunication company, take PALTEL as an example. The customer will call (144) to ask for the phone number. He will wait for an operator to reply. The operator will ask him about the name and address. Then if the name is found , He has to record it on a paper or on any something else.

Scenario three: suppose that the user has a computer and a phone directory software installed on it. He has to run his computer . Waits for the operating system and the programs to start. Then he has to start the program and search for the phone number.

Now , in case of using the phone directory system . The user has to enter a name and press the search button . The result will be retrieved in moments.

From the previous analysis , it is obvious that the use of the phone directory system is easier, more comfortable and quicker than the previous three scenarios. It always provides an up to date results while Scenarios one and three don't do.

From the telecommunication companies point of view . Companies take advantage of saving money paid as salaries for operators. The server holding the phone directory is existed in the two cases ; the case of using the phone directory system for searching and the case of employing operators to reply on the users requests . But , in the first case there is no need for operators . So , companies can provide their customers with PDD and there will be no need for operators to reply on users.

1.3 Project Objectives

The project has the following objectives:

1. Provide an easy system as a phone directory system that saves time and effort.
2. To provide an always up to date phone directory system.
3. To provide a friendly user interface.
4. Provide a system that Telecommunication Companies can take advantage.

1.4 Literature Review

There is no hardware projects design as a phone directory, all previous projects are software projects.

These projects have represents the phone directory in many ways:

1. Design a special web sites for phone directory querying like http://tel.search.ch/, this site provides many services related to phone directory.
2. Electronic Telephone Directory: this available as a program that can be installed on the PC and keeps up to date like http://www.theelectronictelephonedirectory.com/electronic-telephone-directory.htm.

So PDS project is the first project that provides hardware phone directory.

1.5 Time Plan

The time plan views the stages of designing and implementing the system components. Table 1-1 and Table 1-2 show the tasks scheduled for the first and second semesters.

Table 1-1: Time Planning (first semester)

Task/Week	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Project Determination			■	■												
Data Collection				■	■	■	■	■	■	■	■					
Studying System Requirements						■	■	■	■	■	■	■				
Design and Analysis							■	■	■	■	■	■	■	■		
Documentation						■	■	■	■	■	■	■	■	■	■	

Table 1-2: Time Planning (second semester)

Task/Week	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Initial Test for the Dev. Kit																
Keypad Interfacing																
Keypad Testing																
LCD Programming																
LCD Testing																
Writing the Operating Program																
Remote Server Implementation																
System Testing																
Documentation																

1.6 Estimated Cost

This section lists the overall cost of the project , the cost includes the hardware cost , the software cost and the human recourses budget.

Hardware cost: includes the costs of components that are to be used to implement the project. Table 1-3 shows these costs.

Table 1-3: Hardware costs

Component	**Quantity**	**Price $**
Embedded Internet Prototyping board	1	$199
In-Circuit Debugger/Programmer	1	
Powers supply and cables	1	
Keypad	1	$30
The Total Cost	$229	

Software costs: includes the costs of software used to implement the project. Table 1-4 shows there costs .

Table 1-4: Software costs

Component	**Quantity**	**Price $**
Compiler Software (PCWH Software)	1	$475
Website	1	$100
The Total Cost	$575	

Human Recourses Budget :

The system group consists of three undergraduate engineering students , the estimated work cost of them is 2000$ per semester.

1.7 Risk Management

The implementation of any project may face some risks during each stage of the project . This section illustrates what are the risks expected to face the project and what are the solutions for these risks.

1.7.1 Expected Risks

Hardware Risks :

- Availability of the components : some components is not available in Palestine so, they must be purchased from outside . A delivery late is expected. Maybe because a late caused by the carrier or because Palestine is occupied ; the late maybe happen on borders.
- Hardware crash.

Software Risks:

- Undesirable and unexpected software errors.

Human Risks:

- An illness during the stages of the project.
- Unavailability of any team member for any reason.

1.7.2 Risk Avoidance

The following strategies will be taken to avoid risks mentioned in the previous section :

- Ordering the components in early time ,especially those to be purchased from outside.
- Taking care when deal with hardware components and ICs.
- Good estimation of the project requirements and costs.
- Taking care of the team members health and safety during the work.

1.8 Report Roadmap

This report consists of three chapters, the following is a brief description of the topics that are covered in each chapter.

Chapter One : Introduction

Demonstrates an overview about the project, project importance, objectives, time plan, estimated costs and risk management.

Chapter Two: Theoretical Background

Focuses on theories and materials that are related to the project.

Chapter Three : Design Concepts

This chapter describes project objectives in details, introduces the design options, shows the general block diagram.

Chapter Two

Theoretical Background

Chapter Two

Theoretical Background

2.1 Overview

This chapter focuses on theories and materials that are related to the project.

2.2 Protocols and Network layers

TCP/IP is the foundation on which many networks, such as the Internet operate. However, TCP/IP is not just one protocol, but a combination of several protocols stacked on top of each other. Just by the name TCP/IP, it is inferred that there is a TCP protocol that operates on top of the IP protocol. Below is a list of network layers and the used protocols in each layer:

Network Layers

Application
Transport & Session
Network
Data Link
Physical Link

Figure 2-1: Network layers and protocols

2.2.1 Network layers

- The physical link defines the physical connection and its properties. Common properties are cable type, data rates, max distances, and so on. When using a modem to dial an Internet Service Provider (ISP), the physical link is phone line.
- The data link defines the protocol used to maintain the physical link, which may include: Data framing and collision detection. The data link layer is divided into two parts, media access control (MAC) and link layer control (LLC). MAC controls access and encodes signals to a valid format. LLC creates a link to the network via low level link negotiation.
- The network layer provides an address and routing information for the packet. The network protocol primarily used on the internet is IP. Addresses are provided in IP via a 4 byte denomination, for example 192.168.100.1. IP can route incoming and outgoing packets by inspecting the IP address in the protocol and determining the best route for such packets.
- The transport and session are actually separate layers; these layers are combined for simplicity. The transport layer provides error checking, error recovery and data flow control. The session layer provides a method for creating a communication session between two points, and may include security and authentication.
- The application layer is the layer at which communication partners are identified, quality of service is identified, user authentication and privacy are considered, and any constraints on data syntax are identified.

When a PC is connected to the Internet via a modem with a PPP connection, all packets originating from that PC will use PPP for the link layer. However, not all machines are connected to the Internet using PPP, so as a packet is sent through the network each unit will use their own link layer protocol.

2.2.2 Protocols

HTTP Protocol

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It can be used for many tasks, such as server names and it provides a set of instructions for accurate information exchange. HTTP protocol is used to allow different computers, terminals to talk with each other. The communication between the *client* (user browser) and the *server* (software located on a remote computer) involves requests sent by the client and responses from the server.

Each client-server transaction, whether a request or a response, consists of three main parts:

1. **A response or request line.**
2. **Header information.**
3. **The body.**

A client can connect to a specific server at port 80 unless it has been changed, and send a request.

TCP Protocol

TCP (Transmission Control Protocol) is a set of rules that are used along with the Internet Protocol (IP) to send data in the form of message units between computers over the internet. TCP is responsible for verifying the correct delivery of data from client to server.

The main characteristics of the TCP protocol are as follows:

- Monitoring the data flow.
- Forming the data in variable length segments.
- Multiplexing the data.
- Control the data speed
- Allows communication to be courteously started and ended.

IP Protocol

The Internet protocol (IP) is the world's most popular protocol because it can be used to communicate across any set of interconnected networks. IP deals with packets that are the most important and fundamental components of the protocol, each packet must have full addressing information.

IP provides several services:

- Addressing: the address consists of 32-bit used to identify the sender and receiver and select the appropriate path for the packet.
- Fragmentation: smaller packets are easy to handle, so IP is used to fragment and reassemble packets.
- Packet timeouts: every IP packet has a field called Time to Live (TTL) which decrements every time a router deals with this packet. If this TTL goes to zero, then the packet is neglected.
- Type of Service: used to specify priorities.
- Source routing: sender of the packet can set its requirements on the packet path.
- Record route: trace the packet path.
- Security: label packets with security features.

2.3 Microcontroller fundamentals

2.3.1 General characteristics of a microcontroller

Microcontroller is a single integrated circuit (IC). It has the basic components of a computer: input, output, memory, arithmetic and logic unit (ALU), and a control unit.

2.3.2 Basic structure of a microcontroller

The four basic parts of a microcontroller are: central processing unit (CPU), internal memory, registers, and input/output (I/O) subsystem. These parts are connected internally through a bus.

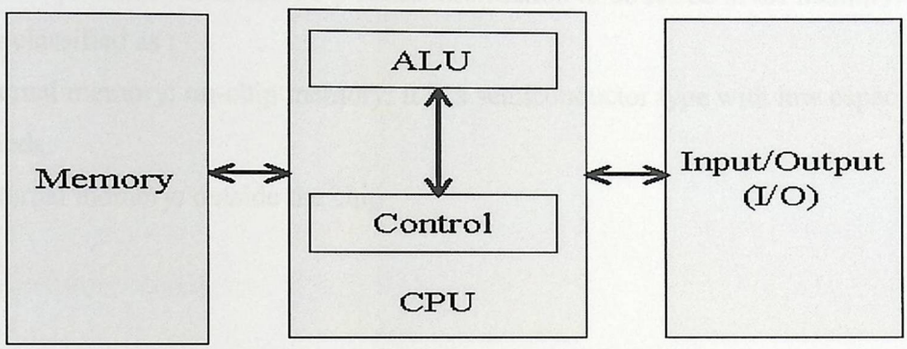


Figure 2-2: A block diagram of a typical microcontroller

I/O Subsystem

The I/O subsystem is used to connect the microcontroller with external parts. The basic component is I/O ports, Each I/O port consist of I/O lines used to transfer information between the external devices and the ports.

Central Processing Unit (CPU)

The CPU of a microcontroller consists of:

1. Arithmetic and logic unit (ALU): performs arithmetic and logic operations.
2. Registers and accumulators.
3. Control unit.

Registers and accumulators are used for temporary storage during CPU operations. The length of each register is equal to the width of the internal data bus.

Control unit is a very important, it is used for monitoring other microcontroller parts. It contains the microcontroller instruction decoding, timing, and control circuitry.

Memory

Memory used to store both data and instructions; dealing with memory can be done using two main operations:

1. Read operation: when the CPU needs the information that is stored in memory.
2. Write operation: when the CPU sends information to be stored in the memory.

Memory is classified as :

1. Internal memory: on-chip memory, it is a semiconductor type with low capacity and high speeds.
2. External memory: outside the chip.

Chapter Three

SYSTEM CONCEPTUAL DESIGN

Chapter Three

SYSTEM CONCEPTUAL DESIGN

3.1 Overview

This chapter focuses on the system objectives, design option and system block diagram.

3.2 Detailed System Objectives

1. Provide an easy system as a phone directory system that saves time and effort. PDS allows the user to get a phone number of someone in few and easy steps.
2. To provide an always up to date phone directory; since the server that holds the database will be in the Telecommunication Companies side.
3. Provide a good user interface, since PDD has a keypad to enter names and to control the system and it has a LCD to view results. It is suitable for people with different educational levels.
4. Provide a system that Telecommunication Companies can take advantage; since the server that holds the phone directory database is existed, but in the case of using PDS there is no need for operators to reply on customers asking for phone numbers.

3.3 Design Options

There are many issues that must be considered in order to build a functional, successful and reliable system. This section demonstrates some options for choosing each part in this project.

3.3.1 The core components options

Option 1:

The first option is to build the system from the basic components and to Interface a MPU with a RAM, EEPROM, LCD, Keypad and network card, or a PIC microcontroller can be used and interfaced with LCD, keypad and network card.

This option requires programming in low level language and it requires wiring or welding the components. So, this option is error prone and needs a lot of time and effort.

The following are two block diagrams of PDD using this option. The first block diagram is in case of using a MPU and the second is in case of using a PIC microcontroller.

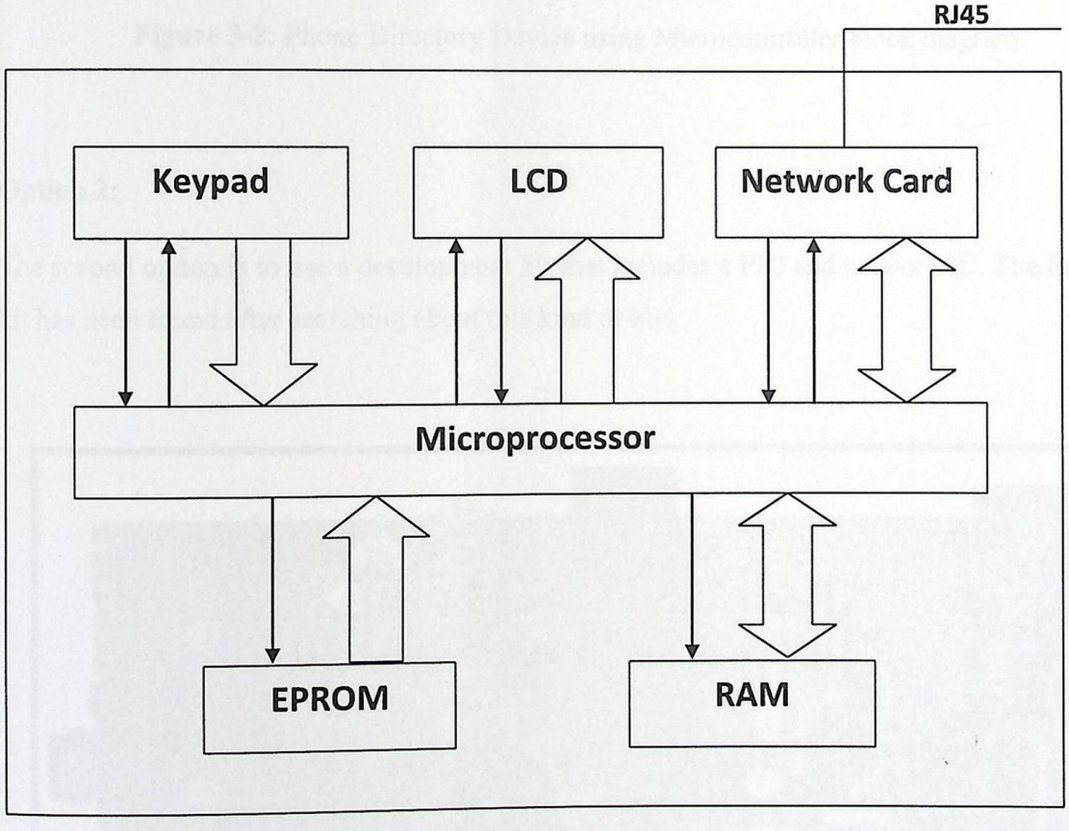


Figure 3-1: Phone Directory Device using MPU block diagram.

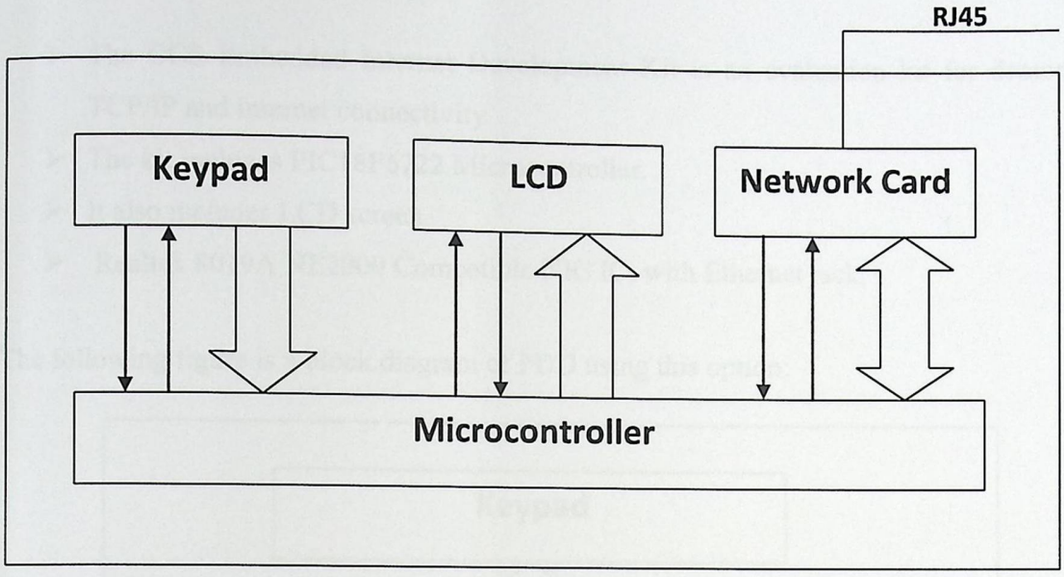


Figure 3-2: Phone Directory Device using Microcontroller block diagram.

Option 2:

The second option is to use a development kit that includes a PIC and network IC. The following kit has been found after searching about this kind of kits.

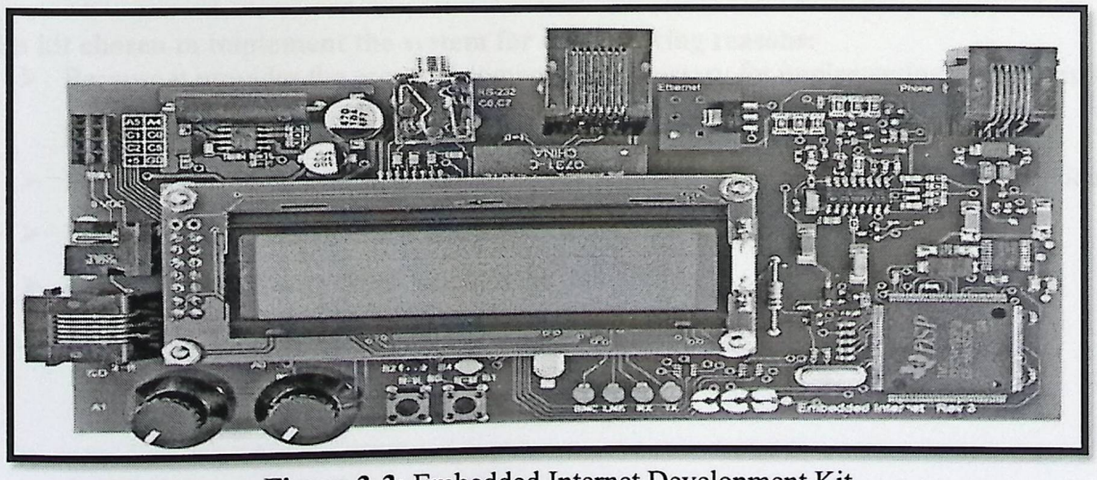


Figure 3-3: Embedded Internet Development Kit

- The CCS Embedded Internet Development Kit is an evaluation kit for demonstrating TCP/IP and internet connectivity.
- The kit includes PIC18F6722 Microcontroller.
- It also includes LCD screen.
- Realtek 8019A NE2000 Compatible NIC IC, with Ethernet jack.

The following figure is a block diagram of PDD using this option:

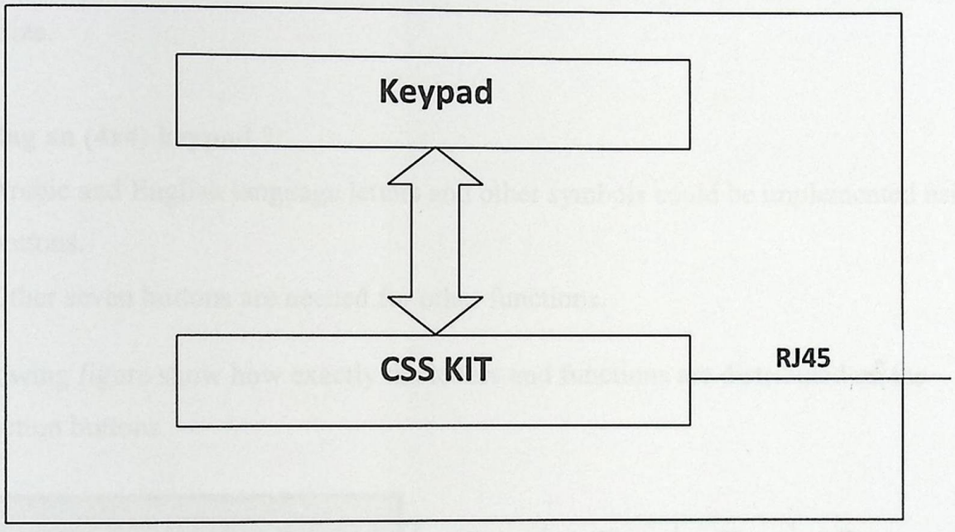


Figure 3-4: Phone Directory Device using Dev. Kit block diagram.

This kit chosen to implement the system for the following reasons:

- Because it provides the essential components necessary for implementing the system; it provides a PIC, LCD and network IC.
- It provides protocols needed for the project; TCP/IP, HTTP, PPP protocols are provided.
- Less expensive than building the system from basic components .
- Easy to interface with other component.
- Helps to implement the system in suitable size.

3.3.2 Peripheral components options

The user can interact with the system through a keypad and LCD.

Keypad Options:

Since the main objective of the system is to search for phone numbers. The user needs to enter a person name to search for his phone number .

To perform this objective a keyboard or a keypad can be used. In this project an (4x4) keypad be used.

The main advantage of choosing a keypad is to implement the phone directory device(PDD) in a suitable size.

Why using an (4x4) keypad ?

- Arabic and English language letters and other symbols could be implemented using nine buttons.
- Other seven buttons are needed for other functions.

The following figure show how exactly the letters and functions are distributed on the multifunction buttons.

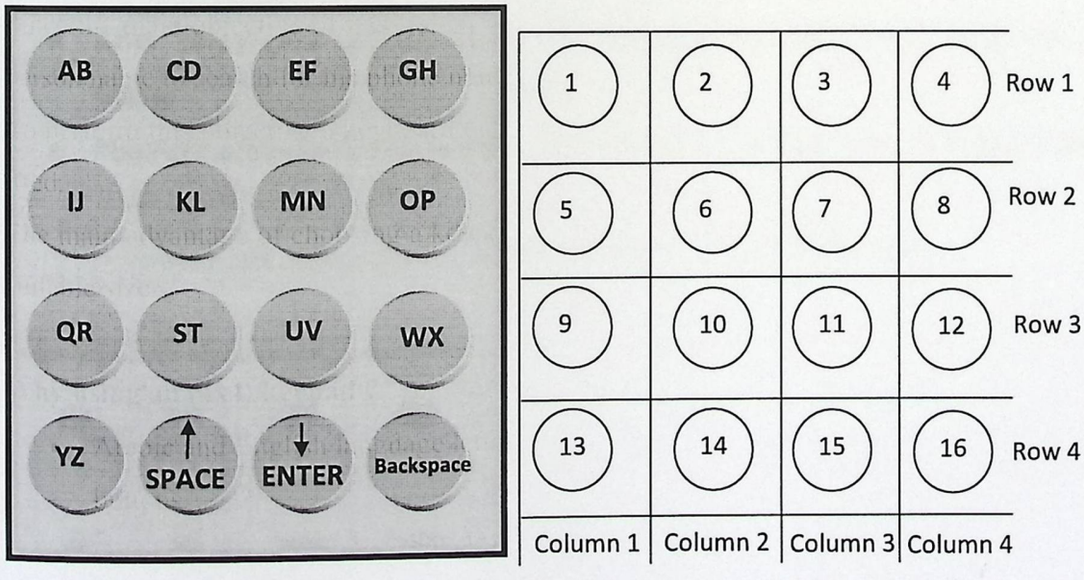


Figure 3-5: Keypad letters and functions distribution

LCD Options:

The chosen Development kit includes a LCD with two lines, the line width is 16 characters. So, there is no need for an external LCD.

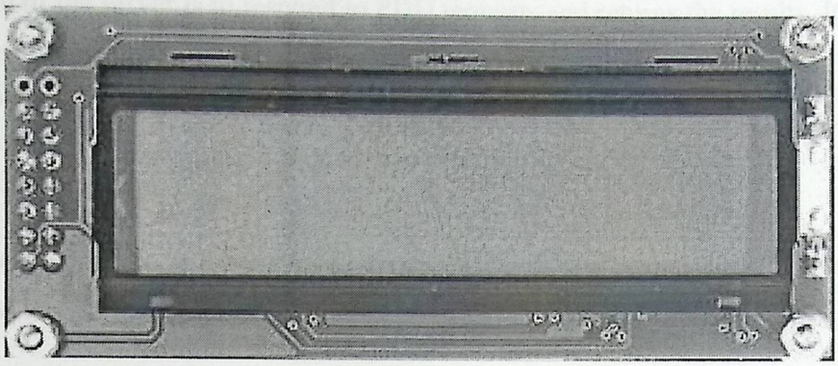


Figure 3-6: LCD

The LCD will be used to view results, status of the system and messages to guide the user. The following are suggested messages to be viewed on the LCD:

- After the user put the device On, the following text with simple animation will be displayed : “Phone Directory System”.
- Messages asking will appear like : “First Name:”, "Second Name", "Third Name:" and "Last Name" .
- A message appearing while connecting with the server: "Connecting".

3.3.3 Server Options

A Windows or Linux web server could be used. A Linux web serveryt used to implement the system. Windows server costs more than Linux servers since Windows operating system and other services are not free , while those for Linux servers are free. An apache web server and MySQL DBMS should be installed on the Linux server.

The main function of the server is to respond to PDD requests and returns the results. It will hold a MySQL database. A server software should be installed on the server to manage it. It will be implemented in PHP programming language.

Database design:

The PDS database keeps track of the phone directory, PDDs, PDDs owners and administrators of the system. The following is a description of the part of PDS to be represented in the database:

1. A phone directory entry consists of first name, second name, third name, address and phone number.
2. Before PDDs can access the server, they should be added to the database. PDD has Serial number and state. Also, all PDDs should have owner.
3. PDDs owners has name, phone number, email and address.
4. All PDDs attempts to access the server are recorded in the database. Each record item has date, IP of the PDD and the name of the person that the search was about.
5. System administrators has username, password, name, email.

The figure in the following page shows the schema of the PDS database using the graphical notation know as **ER diagram**.

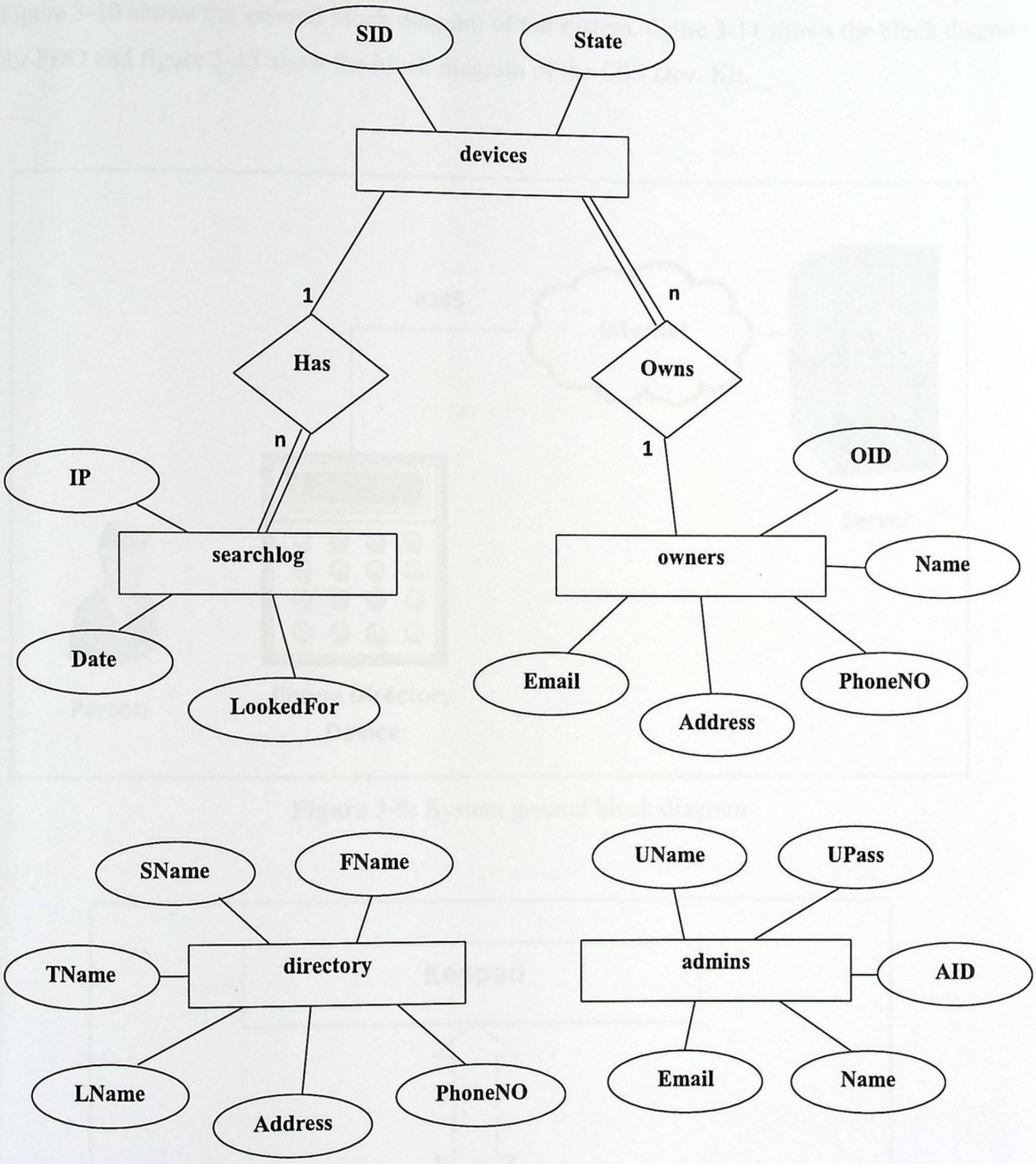


Figure 3-7: EER diagram of PDS database

3.4 System Block Diagram

Figure 3-10 shows the general block diagram of the system, figure 3-11 shows the block diagram for PDD and figure 3-13 show the block diagram of the CSS Dev. Kit.

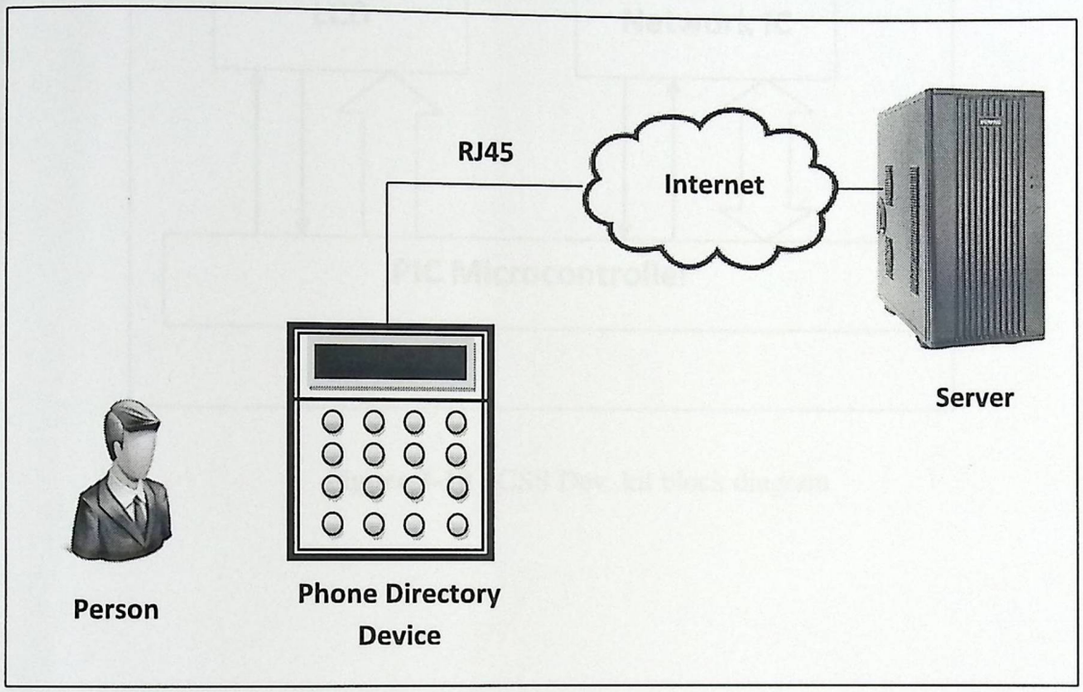


Figure 3-8: System general block diagram

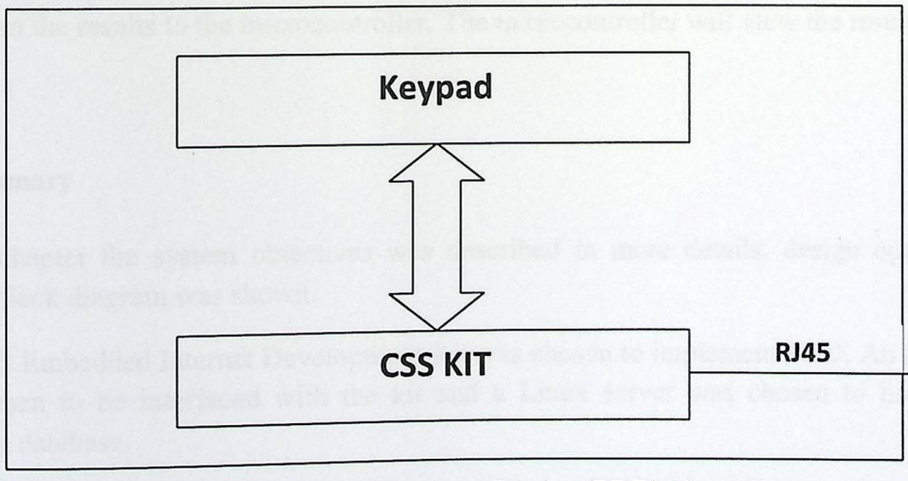


Figure 3-9 : Phone Directory Device (PDD) block diagram

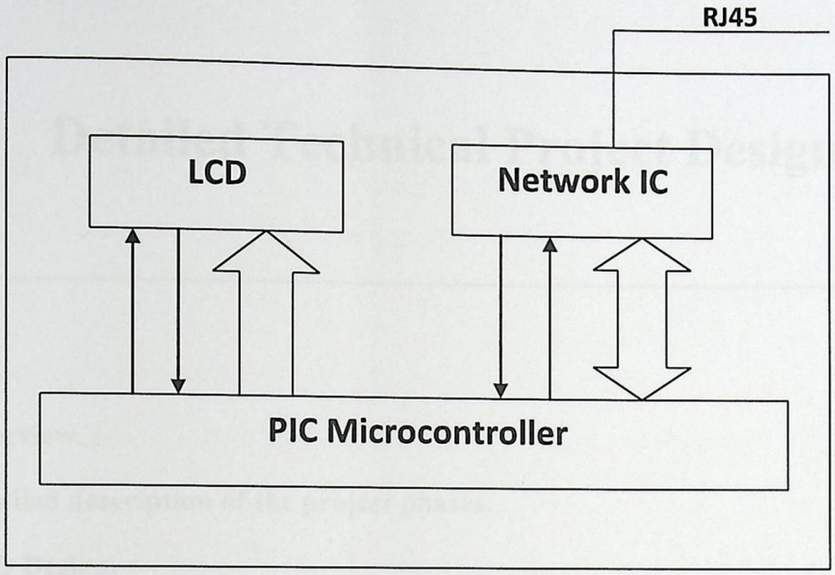


Figure 3-10 : CSS Dev. kit block diagram

As shown in the previous block diagrams. The user has to enter a name using a keypad. Then the microcontroller will send a request to the server. After that the server will respond to the request and return the results to the microcontroller. The microcontroller will view the results on LCD.

3.5 Summary

In this chapter the system objectives was described in more details, design options and the system block diagram was shown.

The CCS Embedded Internet Development Kit was chosen to implement PDD. An (4x4) key pad was chosen to be interfaced with the kit and a Linux server was chosen to hold the phone directory database.

Chapter Four

Detailed Technical Project Design

Chapter four

Detailed Technical Project Design

4.1 Overview.

This chapter focuses on PDD hardware implementation.

4.2 Detailed description of the project phases

The detailed description of the project phases can be summarized as follows:

- Input phase: use a keypad to enter the name and sends it to the PIC as digital value through specific interfacing circuit.
- Processing phase: use PIC (18F6722) as processing unit, its main function is manipulating the inserted data, preparing query and sends it to the server through the established connection, and receive the search results.
- Connection phase: It depends on network IC(Realtek) that establishes a connection to the server.
- Output phase: the results will be displayed on (2x16) LCD.

4.3 PDD Design

This section discusses the schematics, characteristics, feature, and the specifications of each component that are divided into:

- Input system.
- CCS Embedded Internet Development Kit system.

4.3.1 Input System

The main component of the input system is a keypad, so in order to interface it with the CCS Kit, an interfacing circuit is needed.

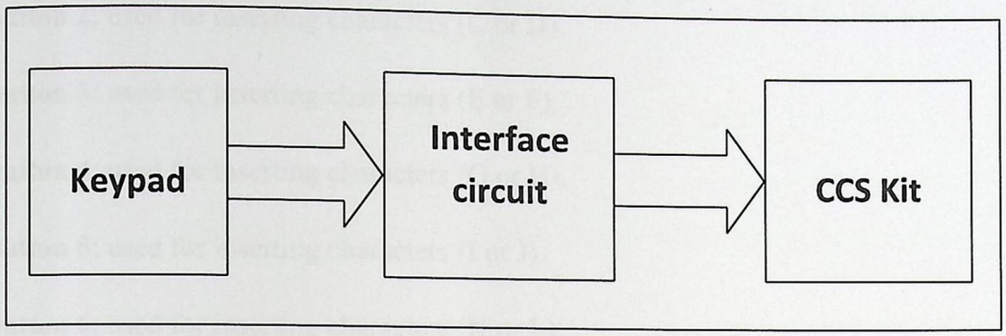


Figure 4-1: Input System

Keypad:

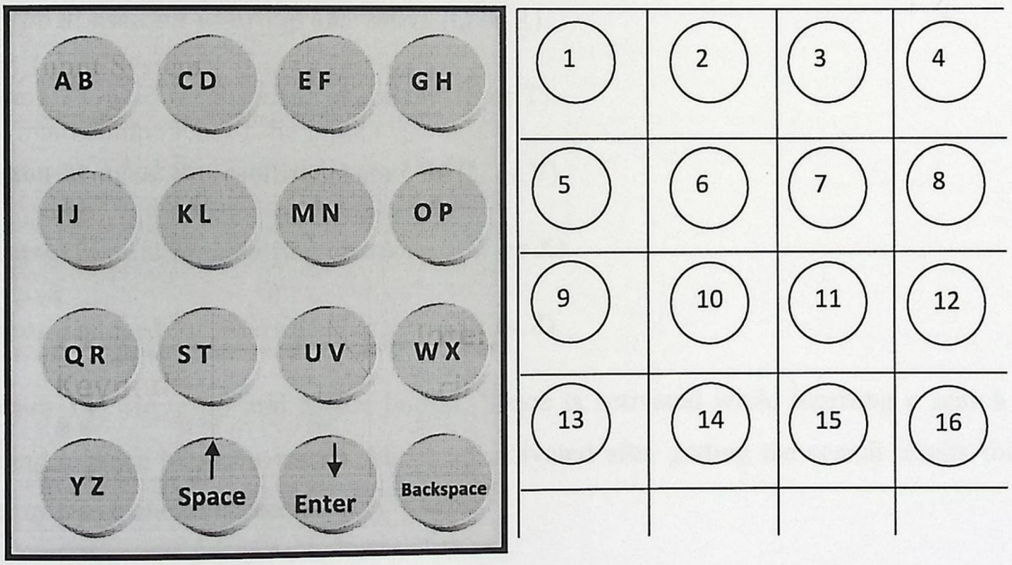


Figure 4-2: Keypad letters and functions distribution

The use of 4x4 keypad stems from its suitability for our project; our keypad contains 16 buttons that are enough for doing all desired functions like:

1. Inserting a person name that user tries to find his phone number.
2. Starting search after insert name.
3. Erase the screen.

Buttons functions:

Button 1: used for inserting characters (A or B).

Button 2: used for inserting characters (C or D).

Button 3: used for inserting characters (E or F).

Button 4: used for inserting characters (G or H).

Button 5: used for inserting characters (I or J).

Button 6: used for inserting characters (K or L).

Button 7: used for inserting characters (M or N).

Button 8: used for inserting characters (O or P).

Button 9: used for inserting characters (Q or R).

Button 10: used for inserting characters (S or T).

Button 11: used for inserting characters (U or V).

Button 12: used for inserting characters (W or X).

Button 13: used for inserting characters (Y or Z).

Button 14: Move up and Space button, Space is activated while inserting a search name for editing a space between words. Move up activated after getting the search results for allowing user to select one of choices.

Button 15: Move down and Enter button, enter is activated while inserting a search name. Move down activated after getting the search results for allowing user to select one of choices.

Button 16: Backspace button, backspace is used to erase one character at a time.

Interfacing circuit:

The interfacing circuit includes the following components:

- Encoder: used to interface a 16 key switch matrix to a digital system. The encoder will convert a key switch to a 4bit nibble.
- Inverter: used to enable output while data are available.

The following figure represents the schematic of the interfacing circuit between the keypad and the kit.

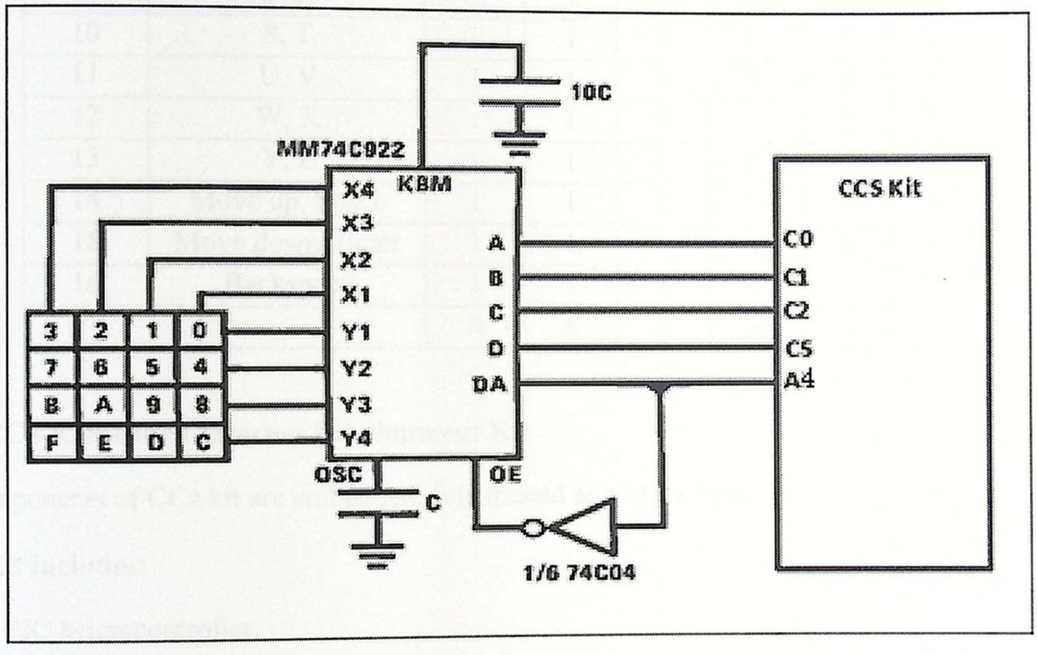


Figure 4-3 Interfacing circuit

The table in the following page shows the keypad functions, and the outputs of the interfacing circuit for each button clicked.

Table 4-1: Truth table of keypad functions and letters

NO.	Function	DA	D	C	B	A	Hex
1	A, B	1	0	0	0	0	0
2	C, D	1	0	0	0	1	1
3	E, F	1	0	0	1	0	2
4	G, H	1	0	0	1	1	3
5	I, J	1	0	1	0	0	4
6	K, L	1	0	1	0	1	5
7	M, N	1	0	1	1	0	6
8	O, P	1	0	1	1	1	7
9	Q, R	1	1	0	0	0	8
10	S, T	1	1	0	0	1	9
11	U, V	1	1	0	1	0	A
12	W, X	1	1	0	1	1	B
13	Y, Z	1	1	1	0	0	C
14	Move up, Space	1	1	1	0	1	D
15	Move down, Enter	1	1	1	1	0	E
16	Backspace	1	1	1	1	1	F
		0	x	x	x	x	

4.3.2 CCS Embedded Internet Development Kit

All components of CCs kit are embedded. It is treated as a black box.

CCS Kit includes:

1. PIC Microcontroller.
2. Network IC (realtek).
3. LCD.

PIC Microcontroller

The kit has a PIC (18f6722), that performs the following tasks:

- Receiving the user entered data.
- Preparing a query and sending to the server.
- Manipulating the returned results and displaying them on LCD.

- Performing processing and calculations.

Network IC

A Realtek 8019A network IC is used to establish an internet connection.

LCD

The LCD is used to display both the input/output data.

Chapter Five

Software Implementation

5.1 PDD Software.

5.1.1 Overview

5.1.2 Requirements Specification

5.1.3 PDD Software Function

5.2 Server Software.

5.2.1 Overview

5.2.2 Requirements Specification

5.2.3 Functionality

5.2.4 Database Design

5.2.5 Detailed Description

5.2.6 Administration Control Panel

5.2.7 Classes

5.2.8 Graphic User Interface

5.2.9 Validation

5.2.10 Browsers Compatibility

5.2.11 Security

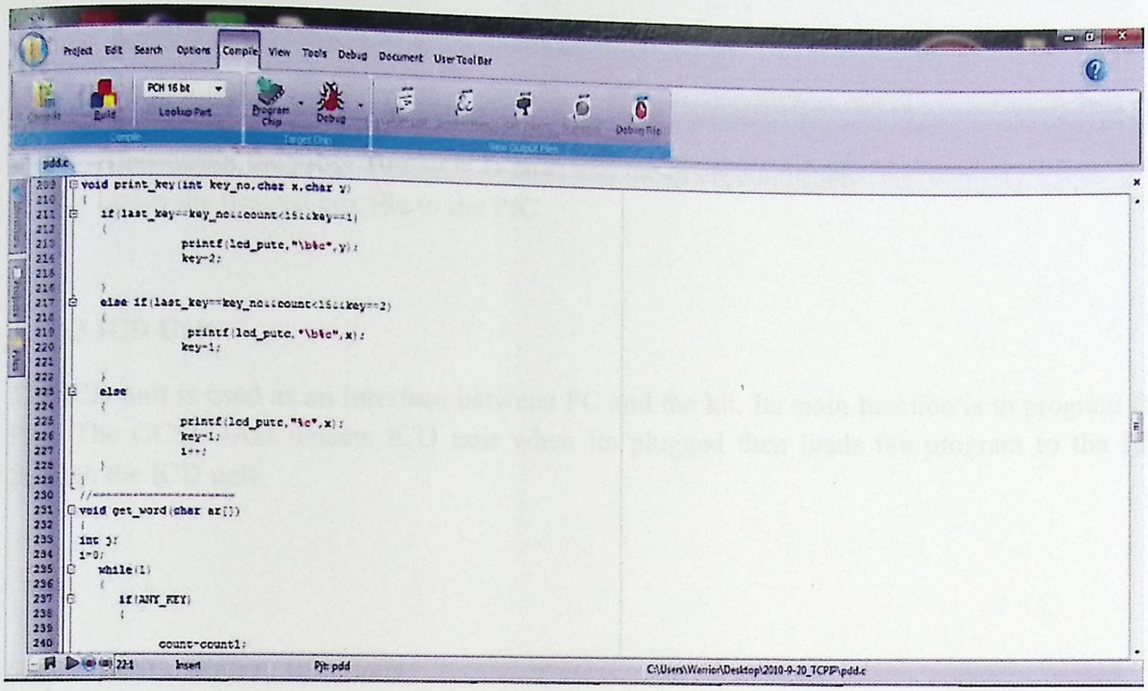


Figure 5-1: PIC C Compiler

5.1.2.2 CCSLoad:

This program used for loading programs to the PIC.

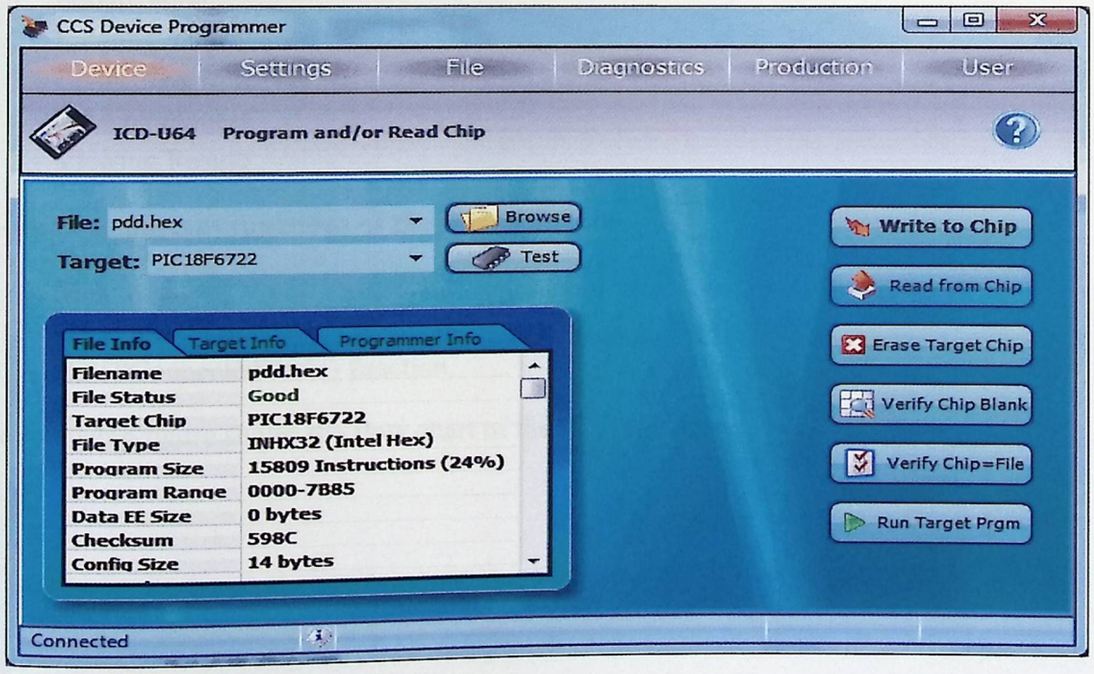


Figure 5-2: CCSLoad

CCS Load features:

1. Auto-detect the programmer(ICD unit) and the microcontroller.
2. Loads the desired hex file to the PIC.

5.1.2.3 ICD Unit

The ICD unit is used as an interface between PC and the kit. Its main function is to program the PIC. The CCSLOAD detects ICD unit when its plugged then loads the program to the PIC through the ICD unit.

5.1.3 PDD software functions

PDD software is an HTTP client that connects to the server. It establishes a TCP connection with the server.

PDD software stages:

1. Initialization.
2. Entering a name.
3. Connection to the server.
4. Sending request.
5. Viewing Results.

The most important functions of the PDD software:

1. Main function.
2. HTTPTask() function.
3. HTTPConnectedTask() function.

The following figure shows the flow chart of the PDD software:

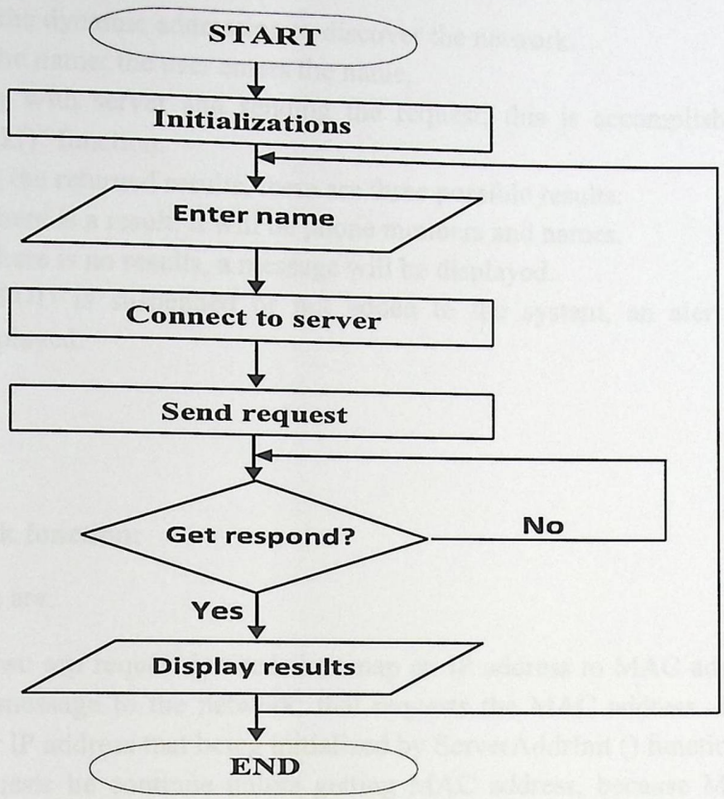


Figure 5-3: PDD software flow chart

5.1.3.1 Main function:

Main function steps are:

1. Initialization : the needed initializations are:
    a. MACAddrInit() : initialization the MAC address, it is a unique and fixed address for each device. This address is important in connection phase because it is one of the device identifiers.
    b. ServerAddrInit() : this function used to initialization server address.
    c. lcd_init() : this function is used to initialize the LCD to be ready for usage.
2. Getting IP address: in this step, the PDD discovers the networks and gets an IP.
   PDD can get an IP address in one of the fowling methods:
    a. Static method: assign a static IP address, this done using IPAddrInit function. This function assigns statically the IP, gateway (GW) and Subnet mask (SM) addresses.
    b. Dynamic method: this method uses the *Dynamic Host Configuration Protocol (DHCP), this protocol used for assigning dynamic IP addresses to the device.* With dynamic addressing, a device can have a different IP address every time it connects to the network.

PDD uses the dynamic addressing to discover the network.

3. Entering the name: the user enters the name.
4. Connecting with server and sending the request; this is accomplished by calling the "HTTPTask()" function.
5. Displaying the returned results, there are three possible results:
    a. If there is a result, it will be phone numbers and names.
    b. If there is no results, a message will be displayed.
    c. If PDD is suspended or not added to the system, an alert message will be displayed.

5.1.3.2 HTTPTask function:

The function steps are:

1. ARP request: arp request is needed to map an IP address to MAC address by sending a broadcast message to the network; that requests the MAC address. This request needs web server IP address that being initialized by ServerAddrInit () function.
   These requests be continue unless getting MAC address, because MAC address is an important part in connection process.
2. Print on the screen the word "connecting", indicates that ARP request completed successfully.
3. Establish the socket (endpoint in a connection). This is the second step in connection. This step requires full remote web server addresses (IP address and MAC address) and a specific port.
   If the socket invalid, print on the screen "socket error", else go to the next step.
4. When the connection process is completed printing on the screen the word "connected". If any problem happened this will lead to force disconnecting.
5. Checking the connection continuously, if it is still connected then starting execution of the HTTPConnectedTask () function, if not, then it must establish the connect again.
6. This step depends on the results of the HTTPConnectedTask () function, if it returns true then disconnect from the server, and if returns false then printing on the screen "Disconnected" and try to connect to the server again.

The figure in the following page shows the flow chart of this function:

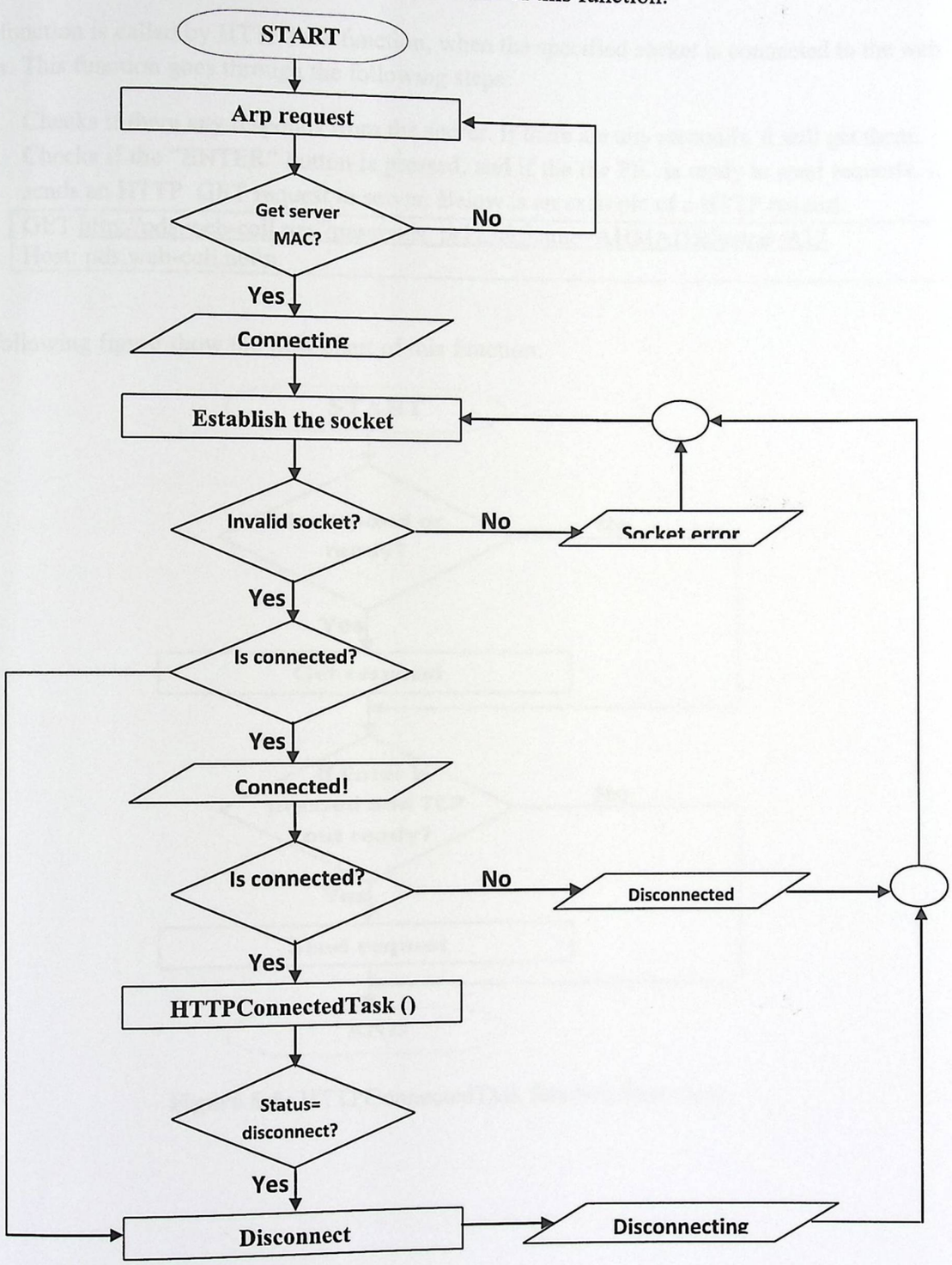


Figure 5-4: TCPTask function flow chart.

5.1.3.3 HTTPConnectedTask function:

This function is called by HTTPTask function, when the specified socket is connected to the web server. This function goes through the following steps:

1. Checks if there any responds from the server. If there are any responds, it will get them.
2. Checks if the "ENTER" button is pressed, and if the the PIC is ready to send requests, it sends an HTTP GET request to server. Below is an example of a HTTP request:

```
GET http://pds.web-cell.net/?pass=pds_pr123&fname=AHMAD&lname=ALI
Host: pds.web-cell.net\n
```

The following figure show the flow chart of this function:

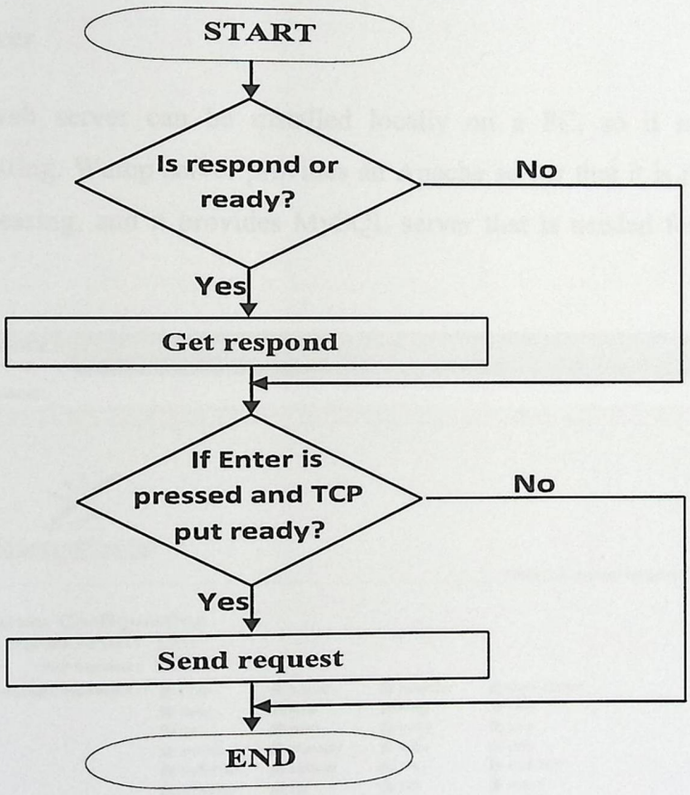


Figure 5-5: HTTPConnectedTask function flow chart.

5.2. Server Software

5.2.1 Overview

This chapter focuses on the server software. The server software is php script that is connected the a MySQL database that holds the phone directory and other information and settings.

5.2.2 Requirements Specification

In order to implement the server software, several programs and tools are required, the following is a description of the needed tools and programs:

- **Wamp Server**

Wamp Server is web server can be installed locally on a PC, so it makes it easier for development and testing. Wamp server provides an Apache server that it is needed for php files compiling and processing, and it provides MySQL server that is needed for implementing the database.

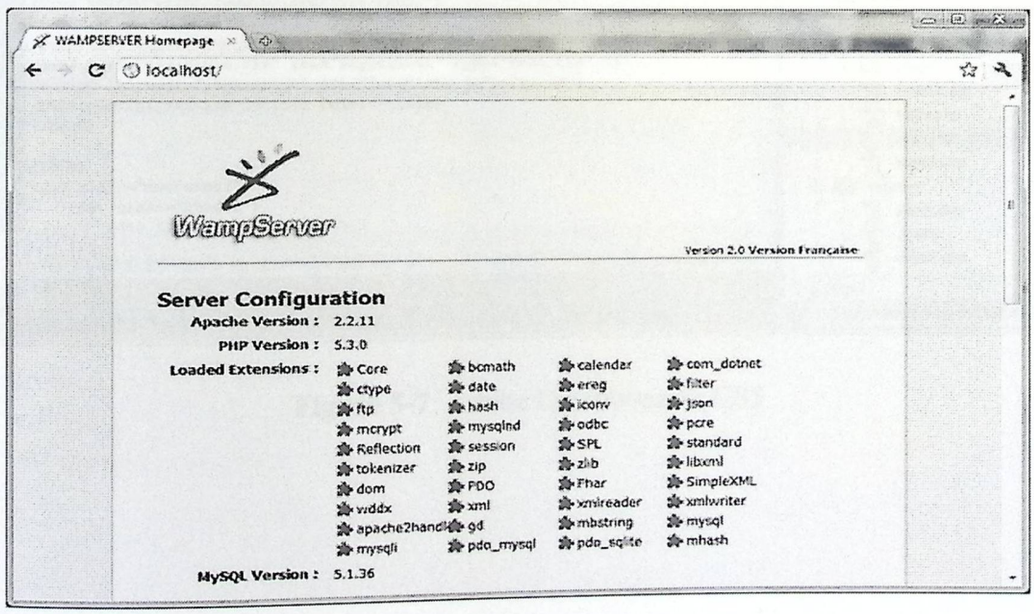


Figure 5-6: Wamp Server index page

- **Adobe DreamWeaver CS5**

Adobe Dream Weaver CS5 is needed as PHP editing program. It provides an easy and flexible environment for development.

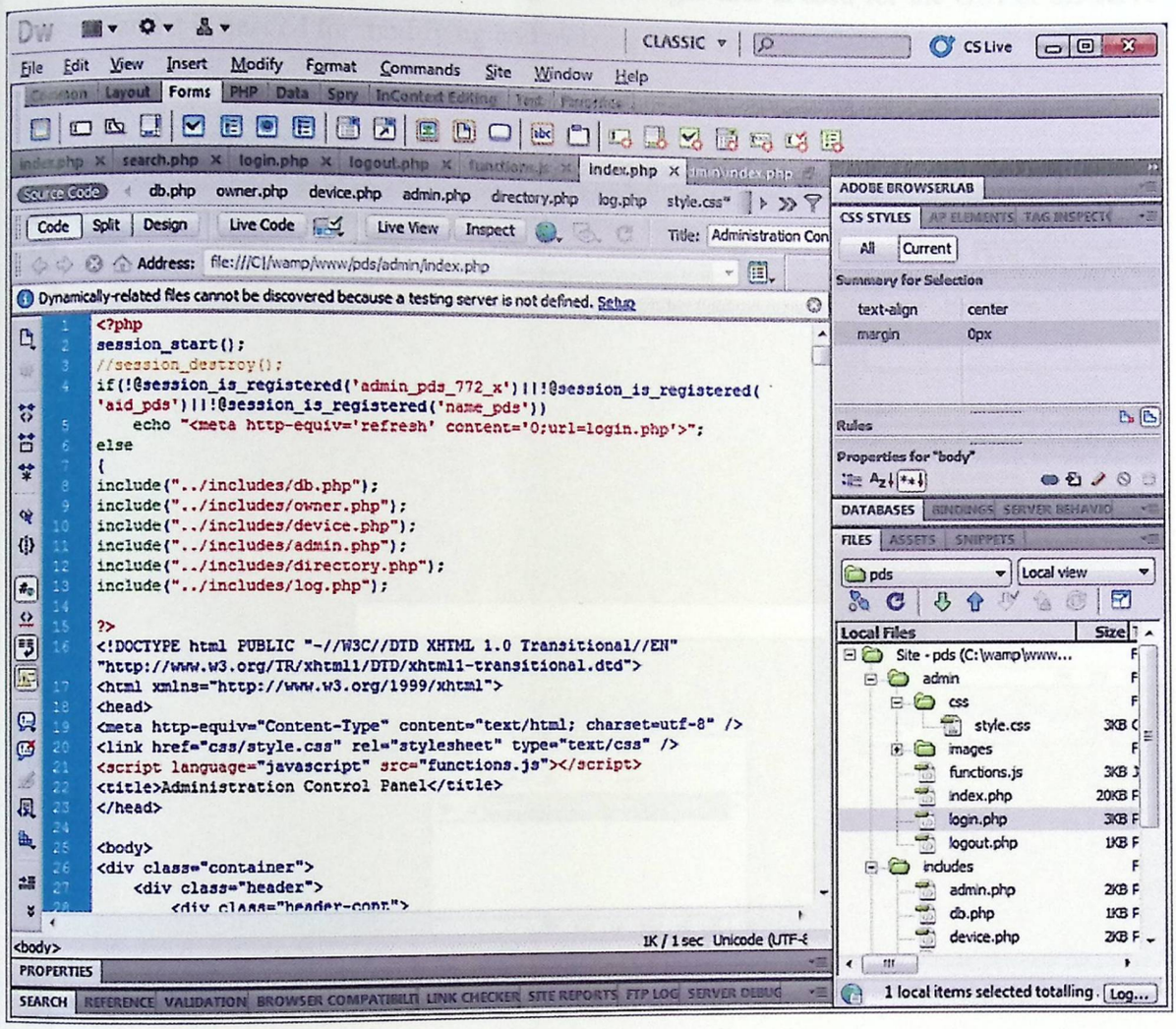


Figure 5-7: Adobe Dreamweaver CS5

- **Adobe Photoshop 8**

Adobe Photoshop 8 is needed for designing some images that is used for the GUI of the server software, and it is needed for modifying and resizing some images.

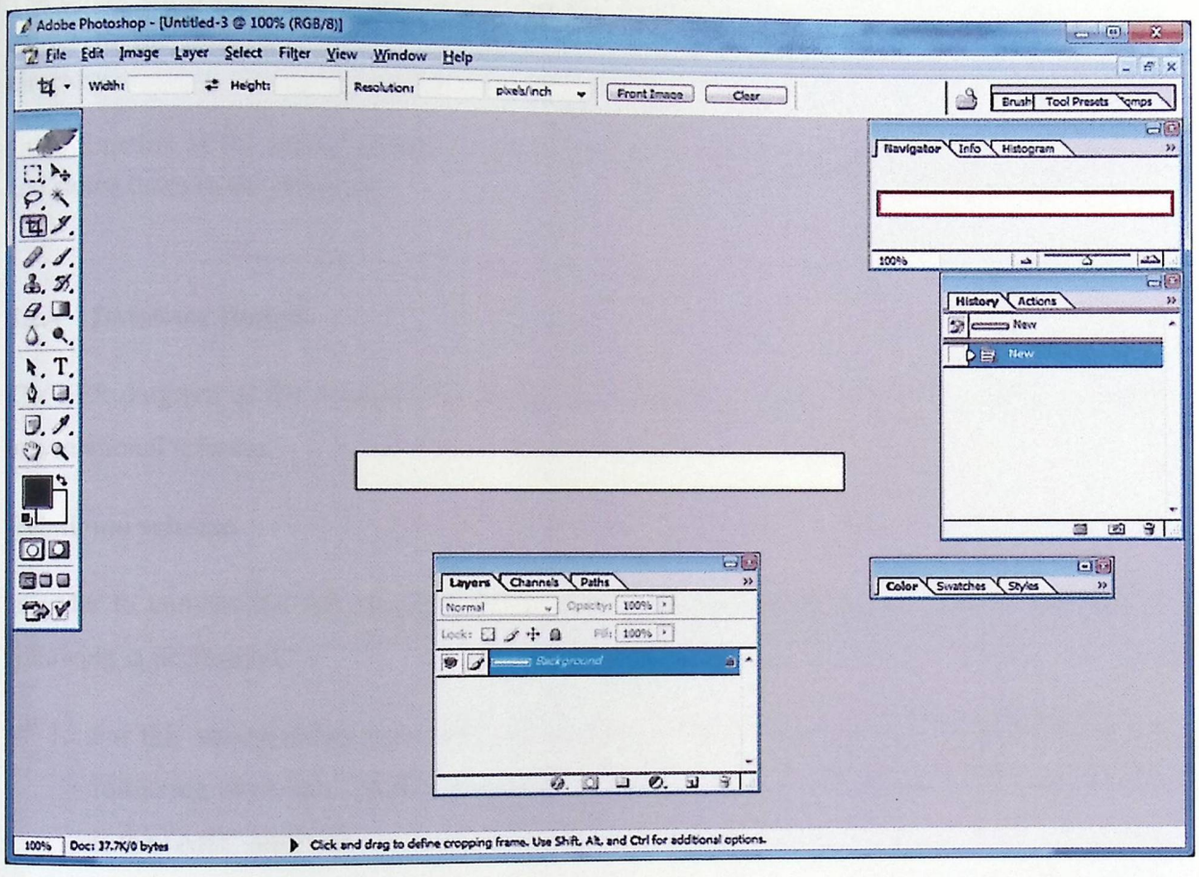


Figure 5-8: Adobe Photoshop 8

5.2.3 Functionality

The main function of the server software is to receive the PDDs requests that asks for phone numbers, and then responds to them. Also, it has other functions, it provides a good user interface to add PDDS to the system; since PDDs can't have access to the server before they have been added. All PDDs should have owners, so owners information should be added too.

The server software make it easy to manage the phone directory. It provides the ability to add entries to the directory. Also it provides the ability to modify and delete entries from the directory.

Other function of the server software is that it records the search attempts for every single PDD, and stores them in the database.

5.2.4 Database Design

The EER diagram of the database was designed in chapter 3. In this chapter it will be converted to a relational schema.

Relational schema

In order to convert the ER diagram shown in the previous figure to a relational scheme, the following is performed:

1. For the strong entity type "admins", a relation "admins" should be created including the following attributes: "AID", "UName", "UPass", "Name" and "Email". The primary key is the "AID" attribute.
2. For the strong entity type "directory", a relation "directory" should be created including the following attributes: "FName", "SName", "TName", "LName", "Address" and "PhoneNo". The "PhoneNo" attribute is the primary key.
3. For the strong entity type "owners", a rlation "owners" should be created including the following attributes: "OID", "Name", "PhoneNo", "Email" and "Address". The primary key of this relation is "OID".
4. For the strong entity type "devices", a realtion "devices" should be created including the following attributes: "SID" and "State". Also the "Owner_ID" attribute should be

included in attributes as a foreign key referring to the owners" relation. The primary key of "devices" consists of "SID" and "Owner_ID" attributes.

5. For the strong entity type "searchlog", a relation "searchlog" should be created including the following attribute: "Date", "IP" and "LookedFor". Also the "SID" attribute should be included as a foreign key referring to the "devices" relation. The primary key of the "searchlog" relation consists of "SID" and "Date" attributes.

The following figure is the relational schema of the database:

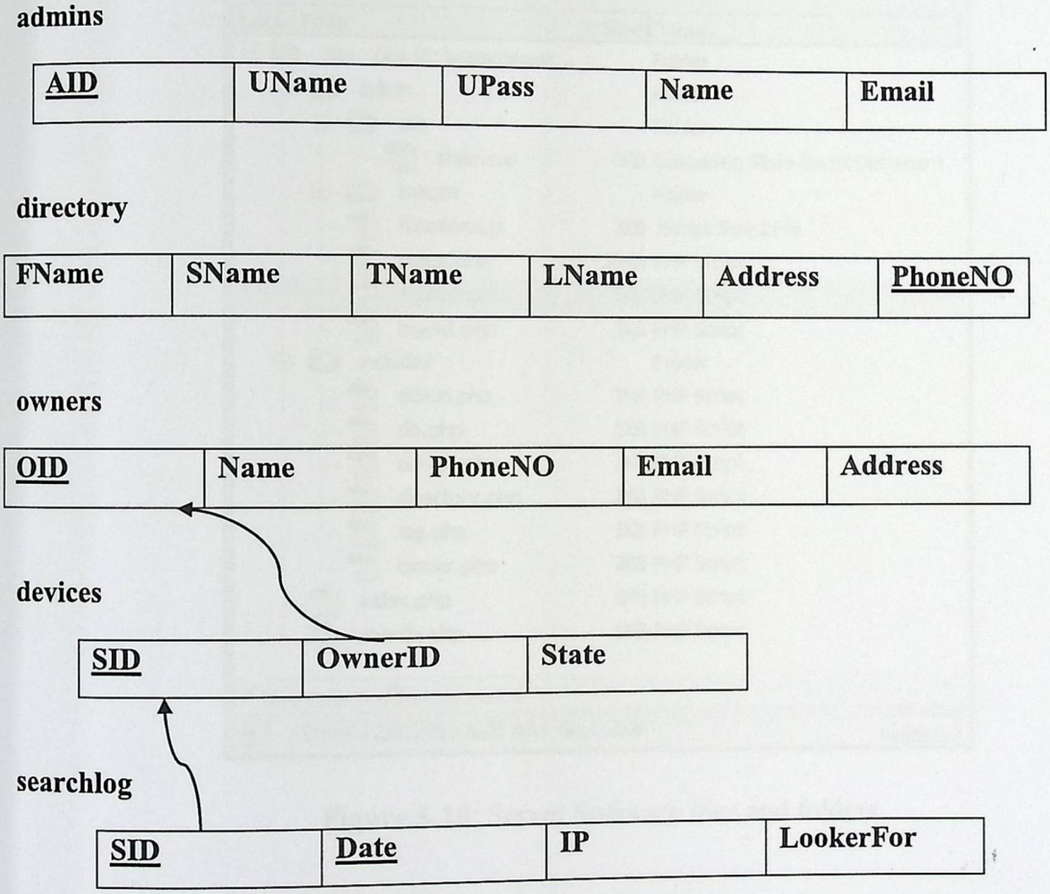


Figure 5-9: Relational schema of PDS database

5.2.5 Detailed Description

The server software has been implemented in PHP programming language along side with HTML, JavaScript and CSS.

The figure below shows the software files and folders as a tree:

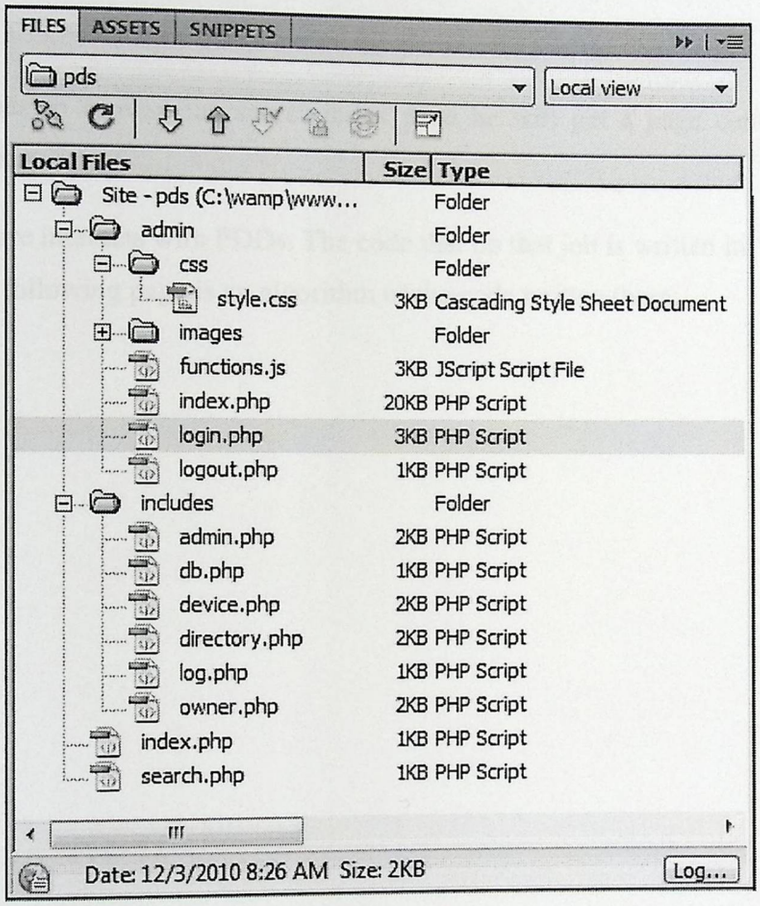


Figure 5-10: Server Software files and folders

As shown in the figure there are two files: "index.php" and "search.php" in the main directory, and the other files are distributed in "includes" folder that holds the classes files, and "admin" folder that holds files related to the system administrators control panel.

The server software has no interaction with end users, so there is no user interface for the public and the "index.php" file only contains the following lines of code:

```
<?php
echo "Restricted Access";
?>
```

So, if anyone tries to browse the server index page he will get a page contains "Restricted Access" as a content.

The server software interacts with PDDs. The code that do that job is written in "search.php" file. The figure in the following page is an algorithm of the code written there:

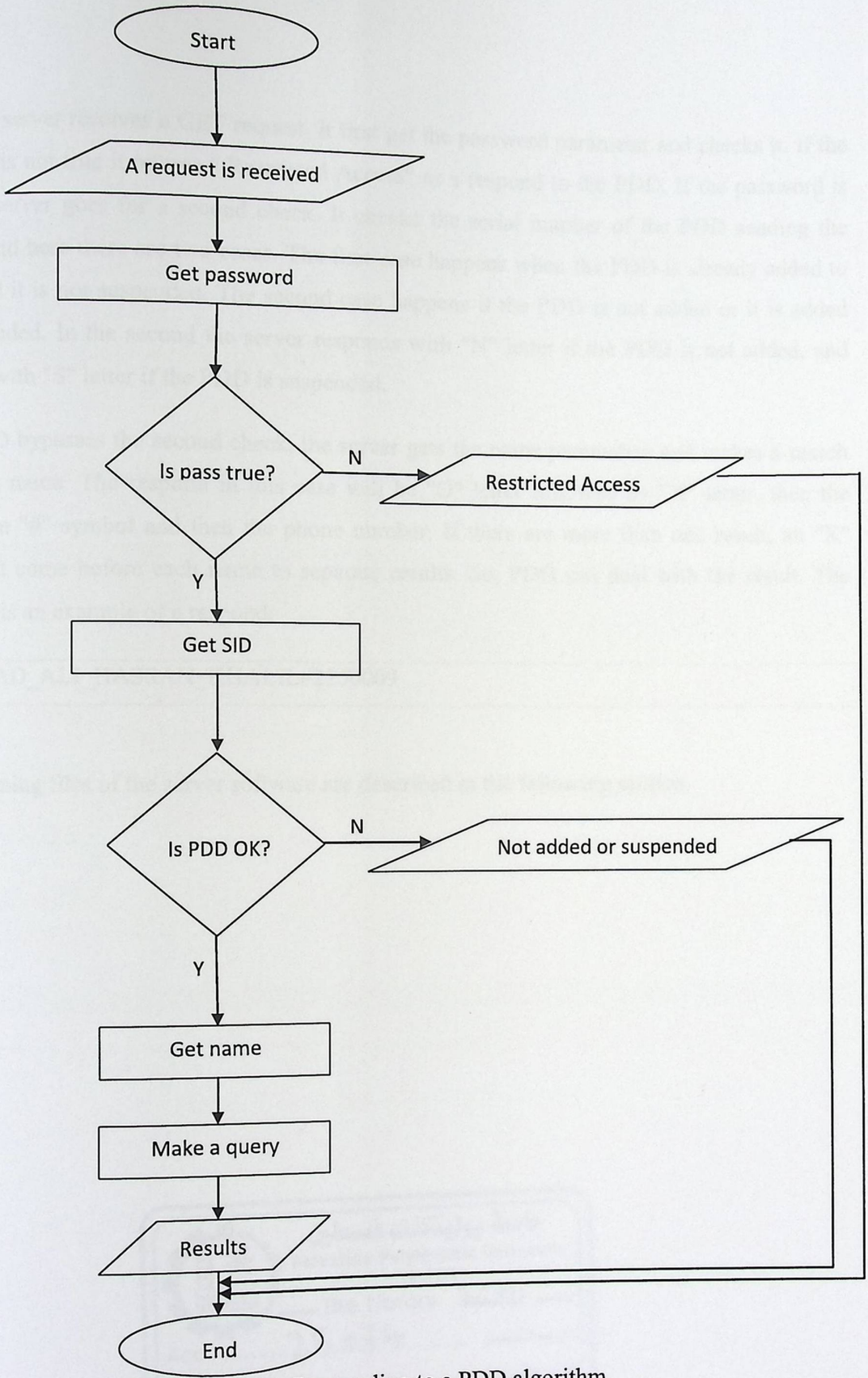


Figure 5-11: Responding to a PDD algorithm

When the server receives a GET request, it first get the password parameter and checks it. If the password is not true it returns " Restricted Access" as a respond to the PDD. If the password is true, the server goes for a second check. It checks the serial number of the PDD sending the request, and here there are two cases. The first case happens when the PDD is already added to server and it is not suspended. The second case happens if the PDD is not added or it is added but suspended. In the second the server responds with "N" letter if the PDD is not added, and responds with "S" letter if the PDD is suspended.

If the PDD bypasses the second check, the server gets the name parameters and makes a search about that name. The respond in this case will be "O" letter followed by "X" letter, then the name, then "#" symbol and then the phone number. If there are more than one result, an "X" letter must come before each name to separate results. So, PDD can deal with the result. The following is an example of a respond:

0XAHMAD_ALI_HASSAN_KHALIL#2250009

The remaining files of the server software are described in the following section.

5.2.6 Administration Control Panel

The control panel files are located in "admin" folder. The file that handles all tasks of the control panel is "index.php" in "admin" folder. Before an administrator can browse that file, he should login. So, if the administrator is not logged in, he will be redirected to "login.php" file. There he can enter his user name and password. If the username and password matches a session will be created and the session variable will be registered.

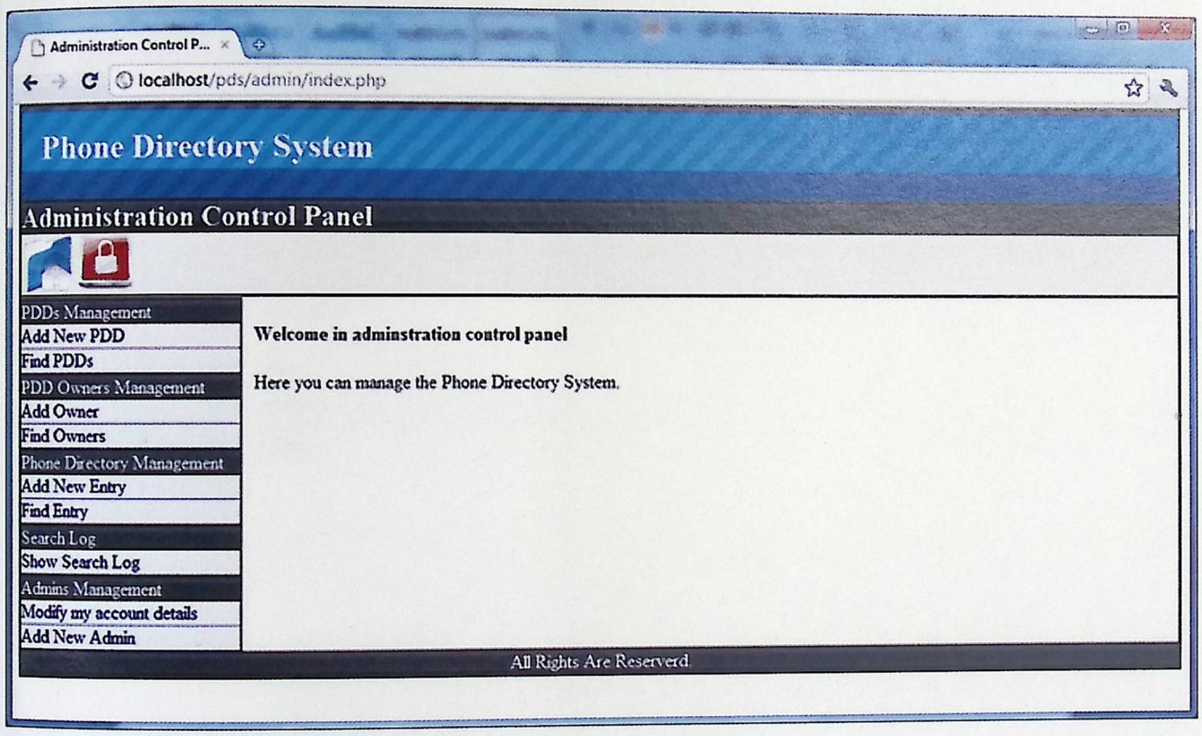


Figure 5-12: Administration control panel

5.2.6.1 PDDs Management

Adding new PDD:

To a new PDD to the system so it will be ready to access the server, the following steps should be performed:

1. Click on "Add New PDD" in the left sidebar, then you will see the following form.

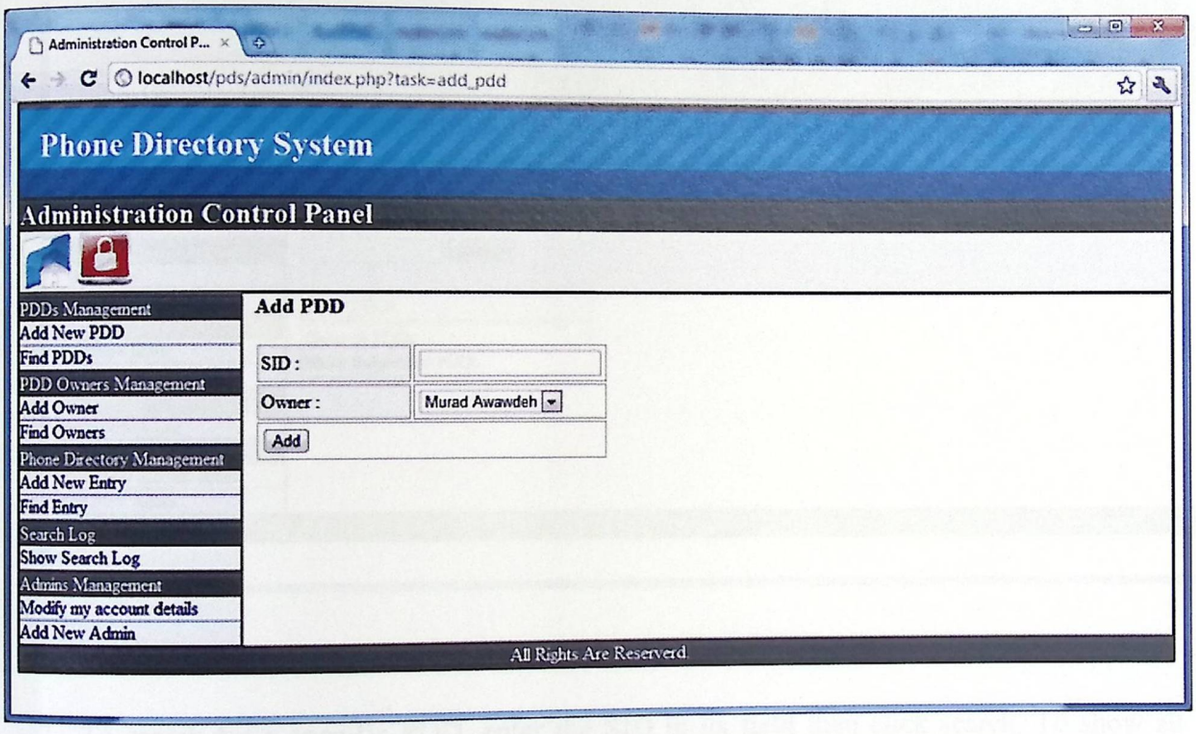


Figure 5-13: Add new PDD form

2. In the SID field, enter the serial number of the PDD.
3. Choose an owner for the PDD.
4. Click on add button.

Finding PDDs:

To show PDDs or to find a specific PDDs, the following steps should be performed:

1. Click on "Find PDDs" in the left sidebar, then you will see the following form.

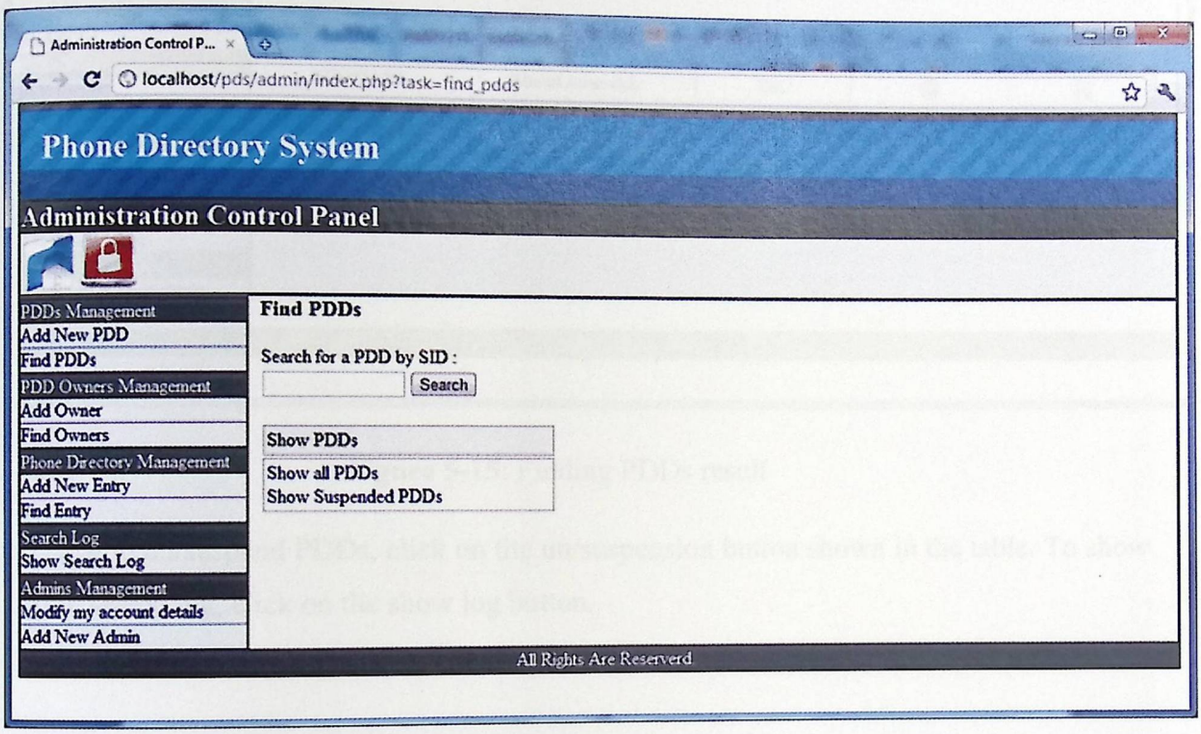


Figure 5-14: Find PDDs form

2. To search for a specific PDD, enter the SID in its field then click search. To show all PDDs click on "Show all PDDS" and to show suspended PDDs click on "Show Suspended PDDs".
3. The result will be as the following figure.

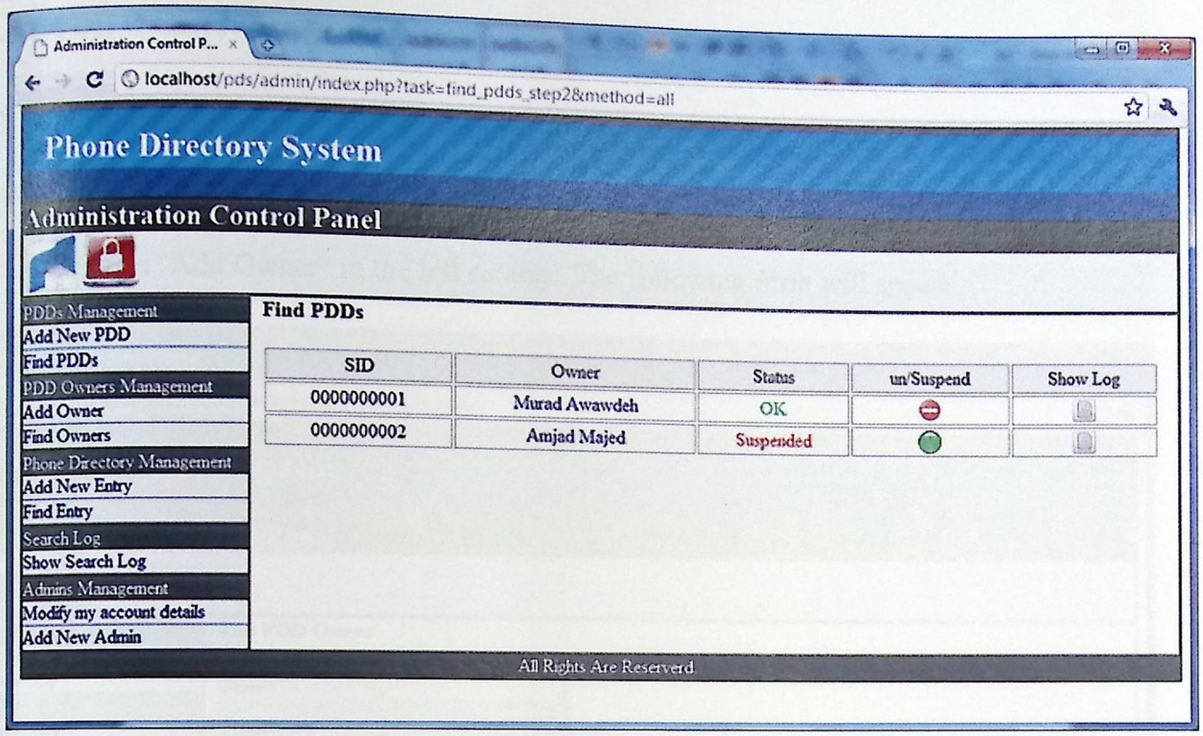


Figure 5-15: Finding PDDs result

4. Now to un/suspend PDDs, click on the un/suspension button shown in the table. To show the search log, click on the show log button.

5.2.6.2 PDD Owners Management

Add owners:

To add a new PDD owner, the following steps should be performed:

1. Click on "Add Owner" in the left sidebar. The following form will appear.

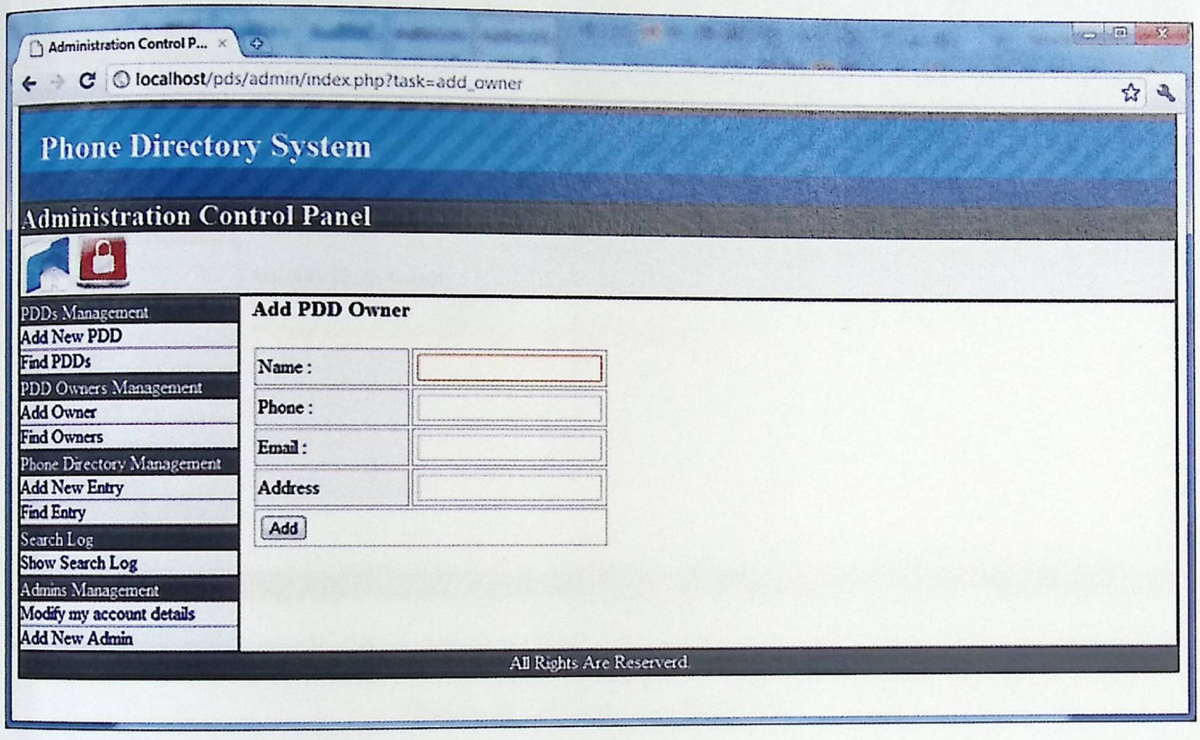


Figure 5-16: Add new owner form

2. Fill the fields, then click the add button.

Finding Owners:

To search for a PDD owner, the following steps should be performed:

1. Click on "Find Owners" in the left side bar, the following form will appear.

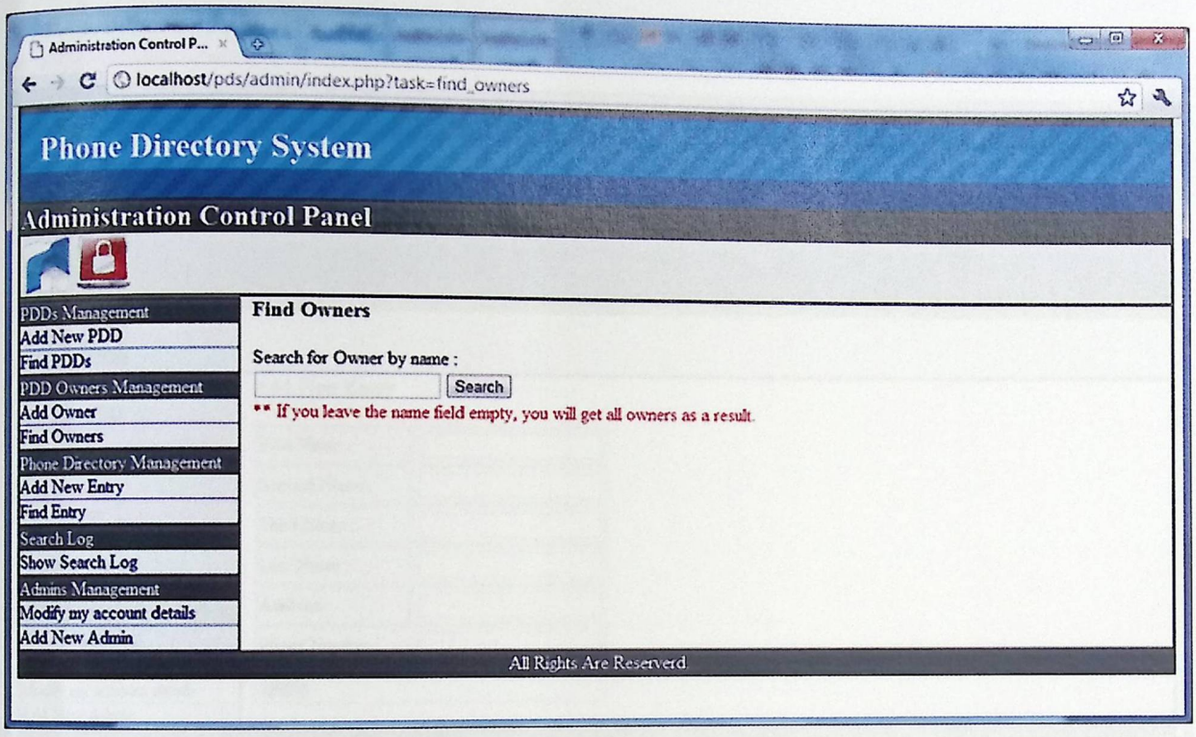


Figure 5-17: Find owners form

2. Fill in the blank or leave it empty to list all owners.
3. Click the search button.

5.2.6.3 Phone Directory Management

Adding an entry to the phone directory:

To add an entry to the phone directory, the following steps should be performed:

1. Click on "Add New Entry" in the left sidebar, the following form will appear.

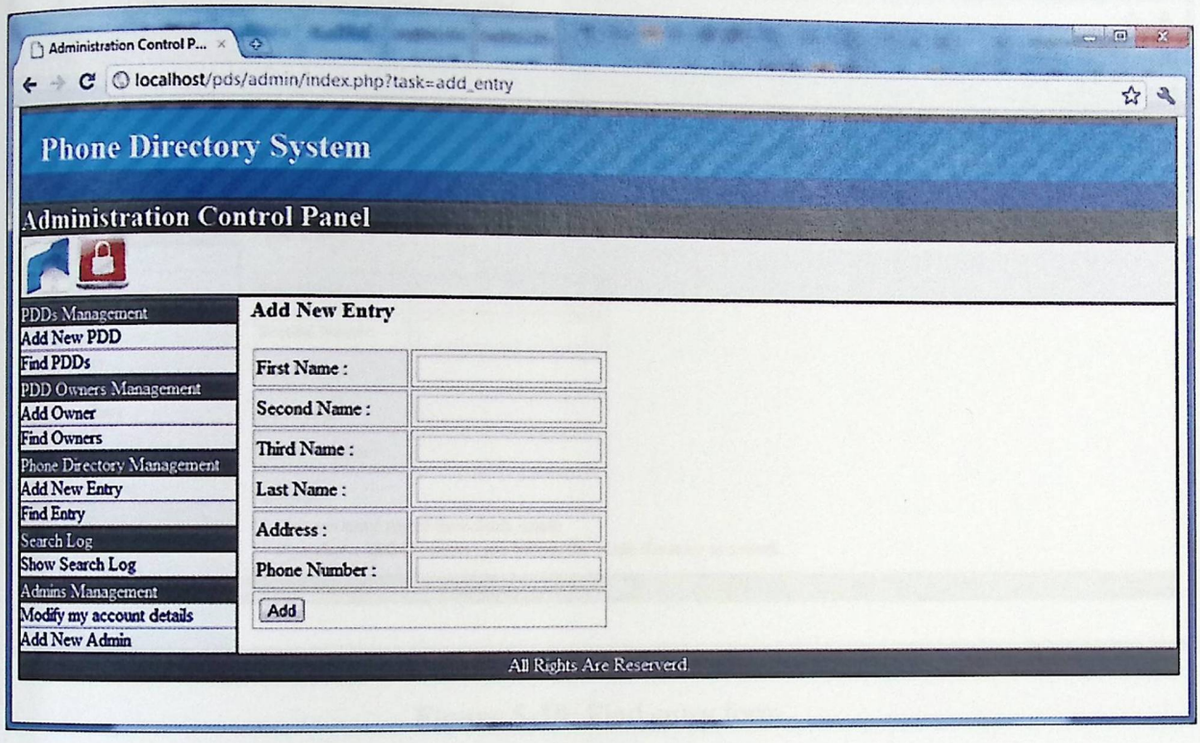


Figure 5-18: Add new entry form

2. Fill in the blanks.
3. Click the add button.

Finding entries:

To search a phone entries, the following steps should be performed:

1. Click on "Find Entry" in the left sidebar, the following form will appear.

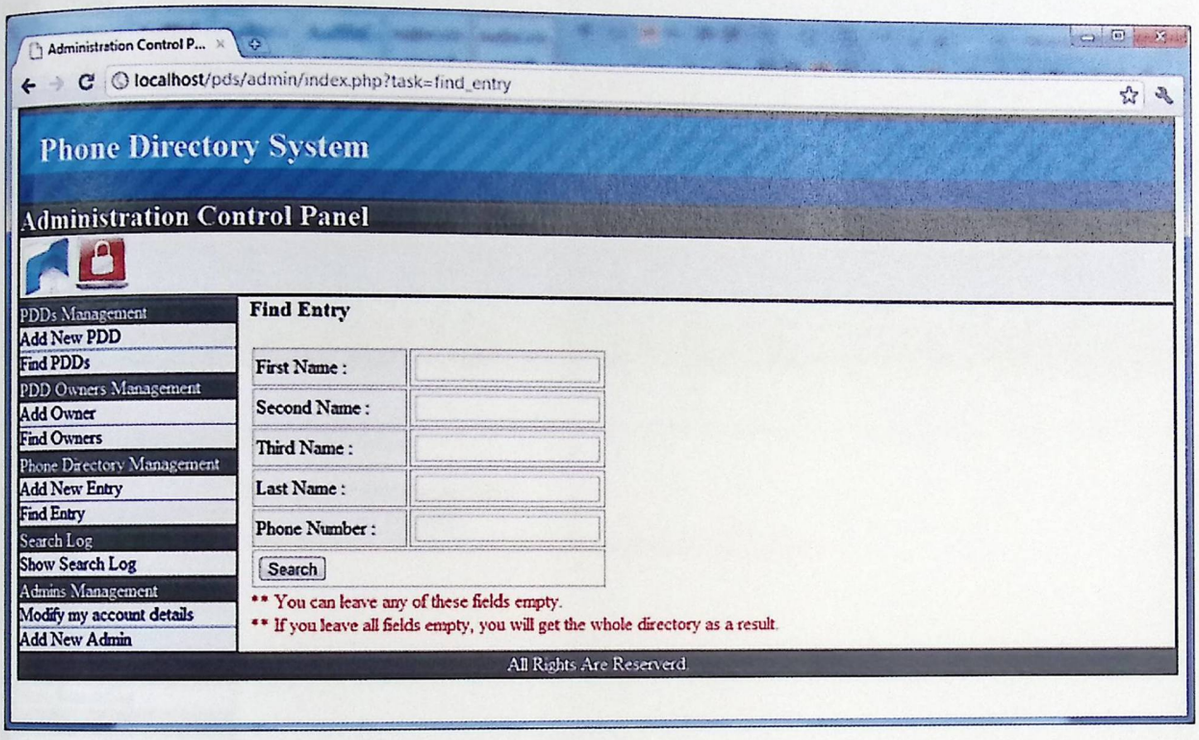


Figure 5-19: Find entry form

2. Fill the needed blanks. You can leave any field empty.
3. Click the search button.

5.2.6.4 Search Log

Showing the search log:

To show the search log for a specific PDD, or for all PDDs, the following steps should be performed:

1. Click on "Show Search Log" in the left sidebar, the following form will appear.

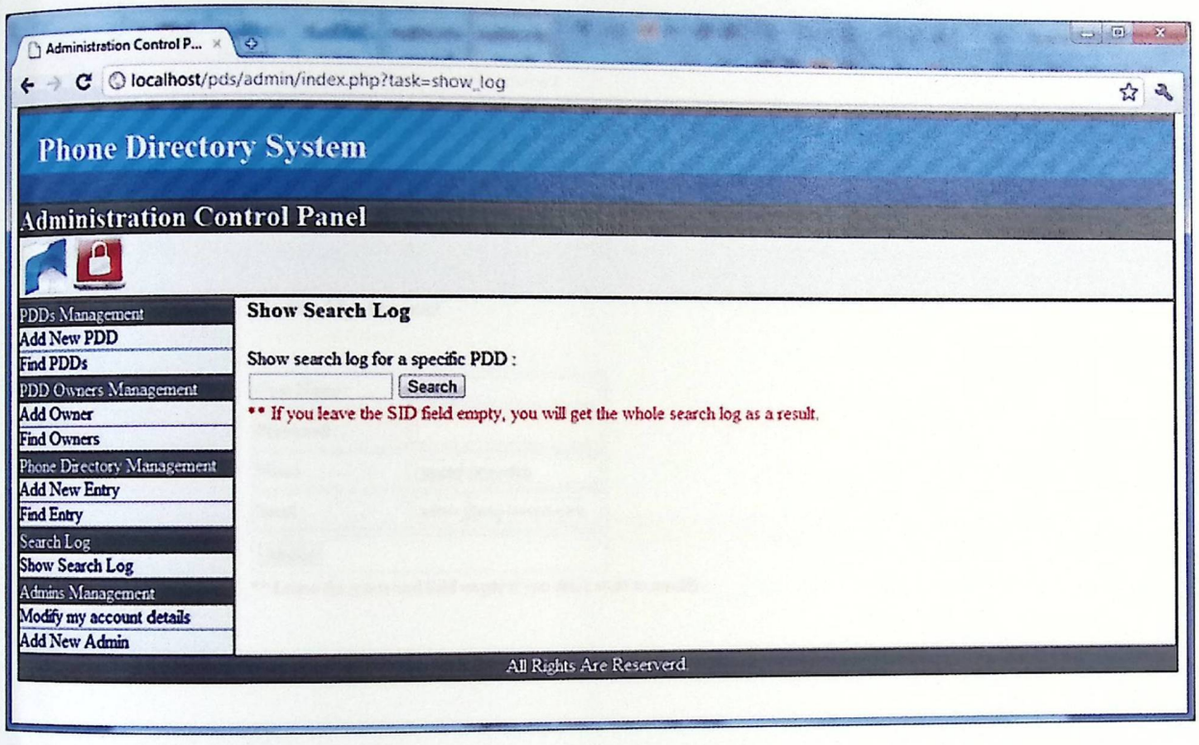


Figure 5-20: Show search log form

2. Fill the blank with a serial number of a PDD, or leave empty to show the whole log.
3. Click the search button.

5.2.6.5 Administrators Management

Modifying account details:

To modify account details for the logged in administrator, the following steps should be followed:

1. Click on "Modify my account details in the left sidebar, the following form will appear.

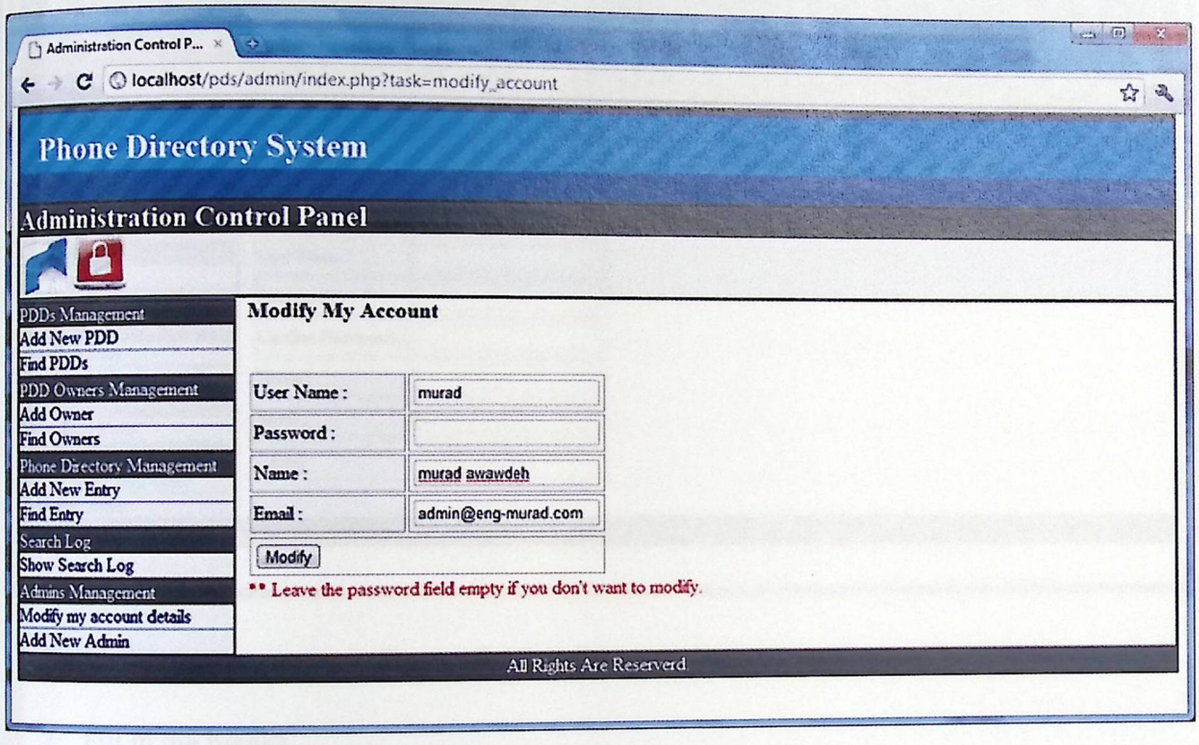


Figure 5-21: Modify account details form

2. The current detail will appear, modify it. The username could not be modified and if the password filed left bank, the password will not changed.
3. Click the modify button.

Adding new administrator:

To add a new administrator, the following steps should be performed:

1. Click on "Add New Admin" in the left sidebar, the following form will appear.

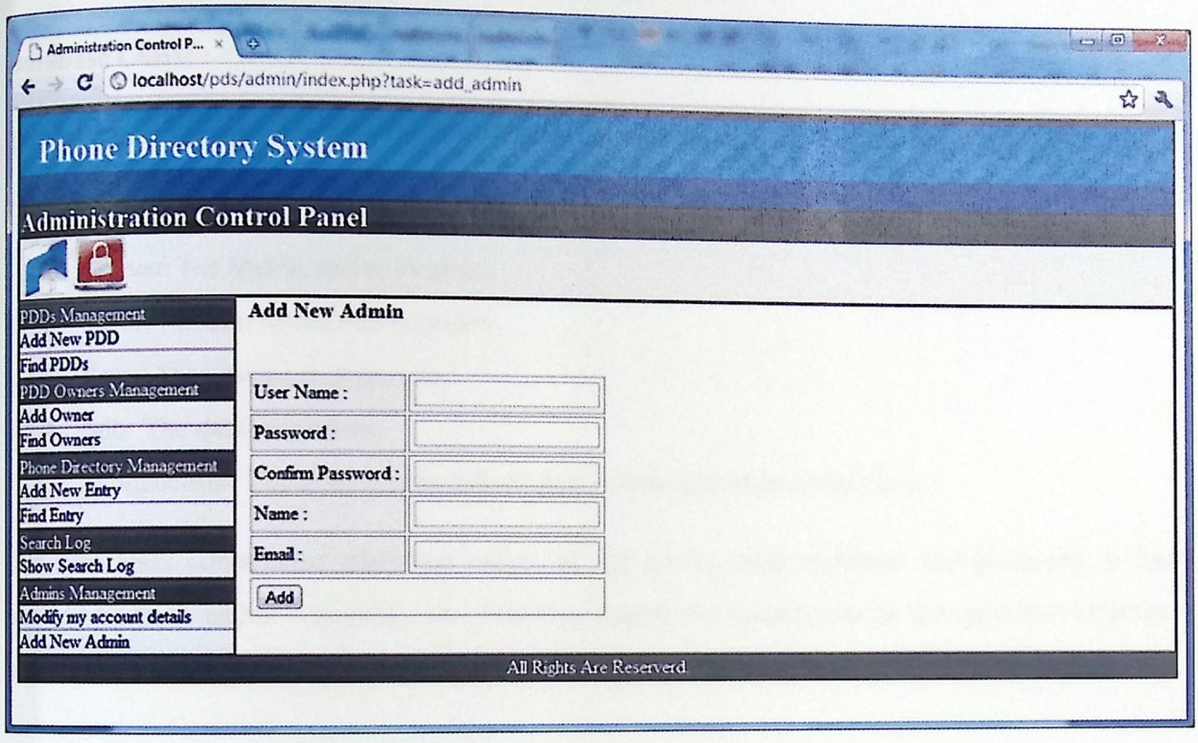


Figure 5-22: Add new admin form

2. Fill in the blanks.
3. Click the add button.

5.2.7 Classes

To achieve the most separation between code and GUI, the majority of PHP code is written in separated classes and each class is in one file. Classes files located in "includes" folder. When this classes is needed, an instances of these classes are created.

Database Class:

The database class name is "db" and it is written in the file "db.php". It is responsible about make connection to PDS database, and it has five parameters:

1. $server: The MySQL server location.
2. $user: The user of the PDS database.
3. $pass: The password of the user.
4. $db: The database name.
5. $connection: The connection stream that is returned after connection.

The "db" class constructor takes the values of the server, user, database and password. It has another function called connect(), this function makes the connection to the specified database and returns the connection stream stored in the variable "$connection".

Administrators Class:

The administrators class named as "Admin" is written in the file "admin.php". This class deals with all actions related to the administrators of the system.

The class parameters:

1. $uname: the user name of an administrator.
2. $name: the administrator name.
3. $email: the email of administrator.
4. $upass: the administrator password.
5. $con: the stream of the database connection.

The class functions:

1. The constructor: instantiate an object of the class and initiate its parameters.
2. Create_Admin(): creates the administrator in the database and return the creation process status.
3. Is_Admin(): checks if the administrator user name and password matches or not, and return the result. This function is used in log in process.
4. Update(): update an administrator information.
5. Show_Admin($aid): shows information for specific administrator identified by administrator ID ($aid).
6. Update($aid,$name,$email): update ans administrator name and email.
7. Modify_Password($aid,$password): updates an administrator password.

PDDs Class:

The PDDs class names as "Device" is written in the file "device.php". It deals with PDDs actions.

The class parameters:

1. $SID: the serial number of a PDD.
2. $OwnerID: the ID of a PDD.
3. $State: the status of a PDD, it is may be "OK" or " Suspended".
4. $con: the stream of the database connection.

The class functions:

1. The constructor: instantiate an object of the class and initiate its parameters.
2. Add(): creates a PDD in the database.
3. Suspend(): suspends a PDD.
4. unSuspend(): cancels suspension.
5. Check(): checks if a PDD is added to system or not, and checks the state of a PDD.
6. Show($SID,$state=""): shows all PDDs, Suspended PDDs or a specific PDD.

Phone directory class:

This class is named as "Ph_Directory" and written in the file "directory.php". It deals with directory related actions.

The class parameters:

1. $fname: the first name of a person.
2. $sname: the second name.
3. $tname: the third name.
4. $lname: the last name.
5. $address: the address of a person.
6. $phoneno: the phone number.
7. $con: the stream of the database connection.

The class functions:

1. The constructor: instantiate an object of the class and initiate its parameters.
2. Add(): creates an directory entry in the database.
3. Perform_Query(): searches for a specific name and returns his phone number. It respons to a PDD asking for phone number.
4. Show(): shows the directory contents.

Search log class:

This class is named as "Log" and written in the file "log.php". It deals with recording search attempts by PDD. It records the serial number of a PDD, its IP, the time of search process and about what the search was.

The class parameters:

1. $SID: the serial number of a PDD.
2. $ip: the IP of a PDD.
3. $lookedfor: the name of a person that a PDD search for his phone number.

4. $con: the stream of the database connection.

The class functions:

1. The constructor: instantiate an object of the class and initiate its parameters.
2. Add(): creates a search log record in the databse.
3. Search($SID): searches about the search log for a specific PDD, or show the whole log.

The owners class:

This class is named as "Owner" and written in the files "owner.php". It deals with PDDs owners actions.

The class parameters:

1. $name: the name of a PDD owner.
2. $phone: the phone number.
3. $email: the email address.
4. $address: the address of the owner.
5. $con: the stream of the database connection.

The class functions:

1. The constructor: instantiate an object of the class and initiate its parameters.
2. Add(): creates an owner in the database.
3. Update($OID): updates an owner name specified by owner ID ($OID).
4. Delete($OID): delete an owner.
5. Search(): search for an owner and retrieve his information.
6. Show($OID=""): shows an owner information or shows all owners information.

5.2.8 Graphic User Interface

The GUI of the control panel was designed using HTML and CSS, with the help of Photoshop to design some photos.

The main component to build the control panel layout is the HTML . The div that contains all other components has the class "container". It is 1000px width and it is centered. The following figure showing the layout of the control panel.

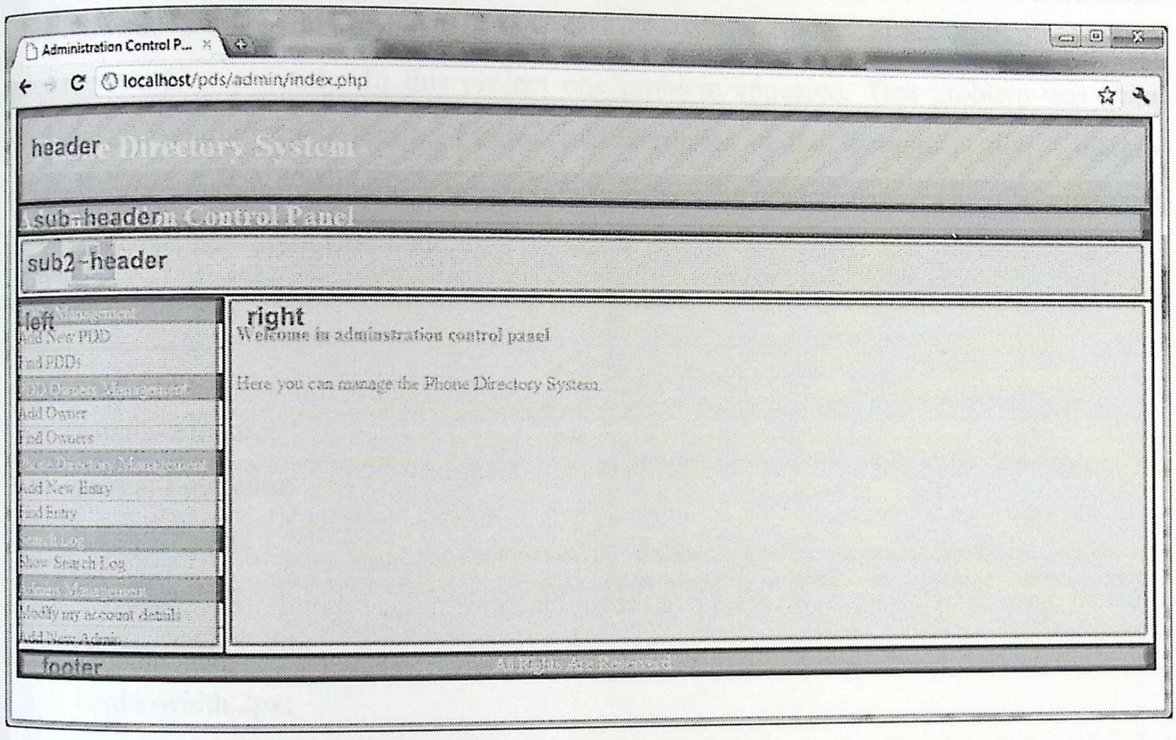


Figure 5-23: Control panel layout

The "sub-header" div contains the title of the control panel. The sub2-header div contains two button; a button to return to the control panel home page, and the other to logout. The "left" dev contains the sidebar. The "right" div contains the contents of pages. Finally, the "footer" div contains the rights text.

5.2.9 Validation

In order to make sure that inputs made by systems administrators are true, validation rules should be performed in forms. JavaScript is the best choice to perform validation. The validation functions are written in "functions.js" file.

5.2.10 Browsers Compatibility

The server software was designed to be compatible with Internet Explorer and Firefox. Since some browsers goes beyond standards, the CSS code must be different for these browsers in order to view files correctly. In this project one problem appeared. That problem was about centering the control panel, for the Firefox two attributes of the "container" div: " margin-left:auto;" and " margin-right:auto;" are enough to center the control panel. The following code shows the "container" class CSS code:

```
.container
{
    margin-left:auto;
    margin-right:auto;
    width:1000px;
    text-align:left;
    border-style:double;
    border-width:2px;
    overflow:hidden;
}
```

But, for Internet Explorer an extra attribute should be added for the "body" tag to center the control panel. That attribute is " text-align:center;". Thus, there is a need to add " text-align:left;" to the attributes of the "container" div to return the text alignment back to left.

The server software was also tested on Google Chrome browser and it was viewed correctly.

5.2.11 Security

Since only the system administrators should access the server, it must be protected from any access attempts. So, the "includes" and "admin" folders are protected by password. If anyone tries to browse these folders, a window appears asking for username and password.

Steps of protecting these folders:

1. Login to the testing website control panel, in this case it is a CPanel control panel.

Figure 5-24: The testing website CPanel

2. Under security, click "Password Protect Ditectories". A window will appear click the go button, as shown in the following figure:

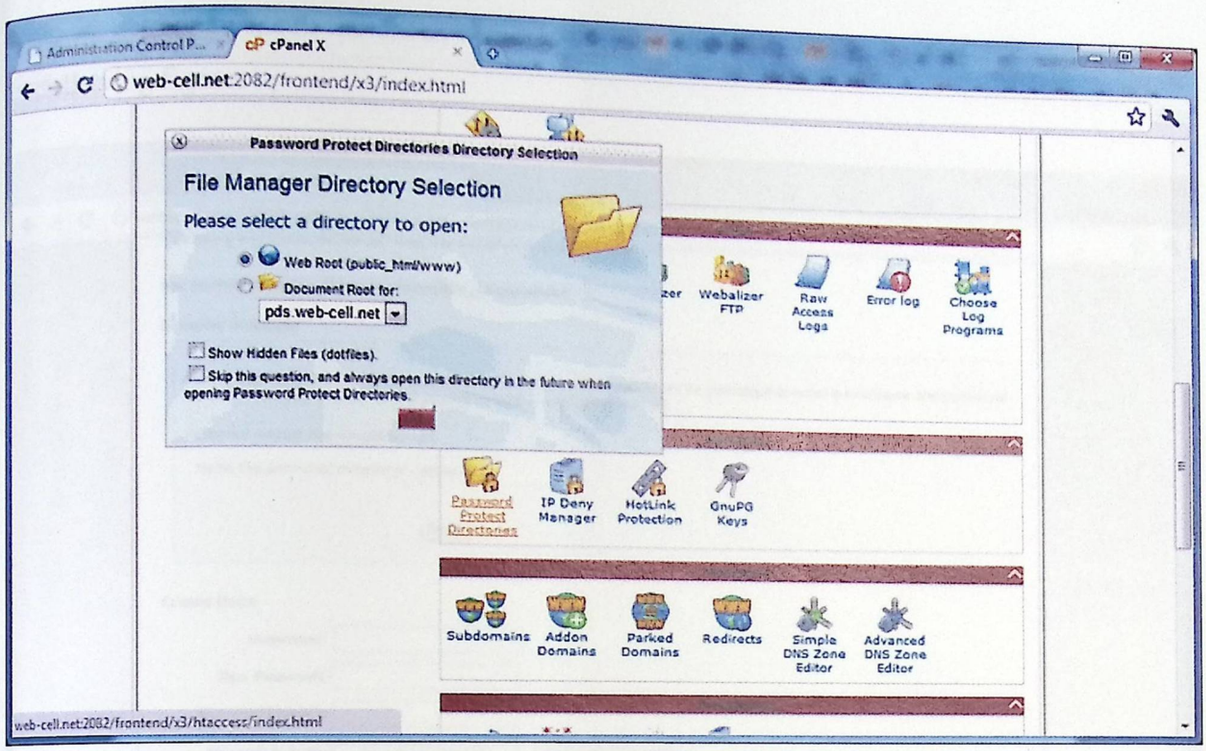


Figure 5-25: Protect directories with password step 2

3. In the next page, click on "admin" directory.

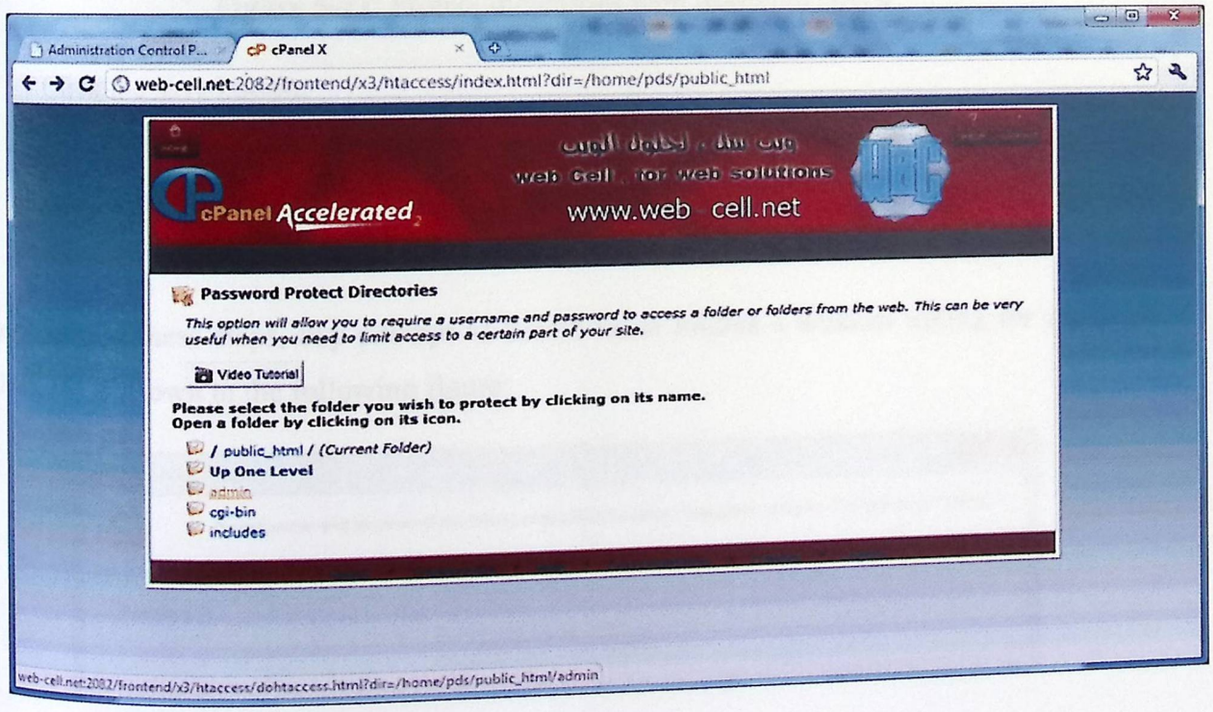


Figure 5-26: Protect directories with password step 3

4. In the next page check the check box and the enter the name of the folder as shown in the following figure:

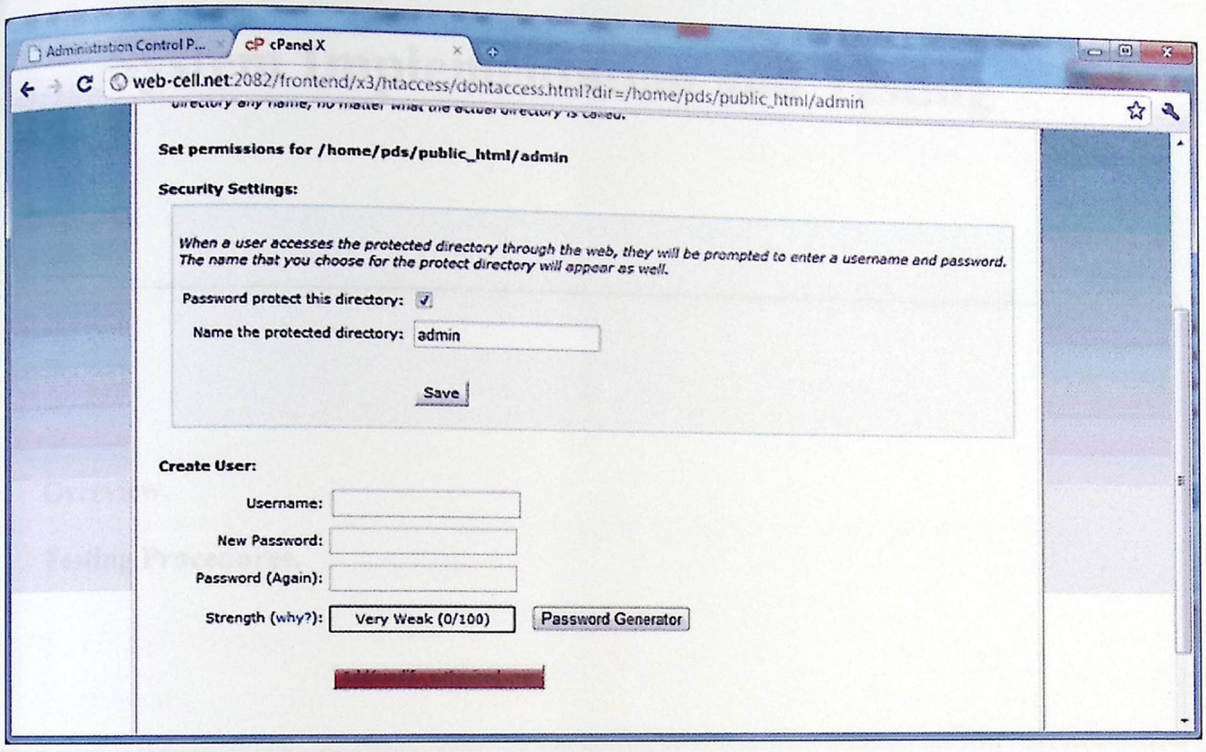


Figure 5-27: Protect directories with password step 4

5. Click the save button.
6. Fill in username, new password, password(again) blanks and click "Add/modify authorized user" button.
7. Repeat these steps for "includes" folder.

After applying these steps, any attempt to access these folders a window asking for username will appear as shown in the following figure:

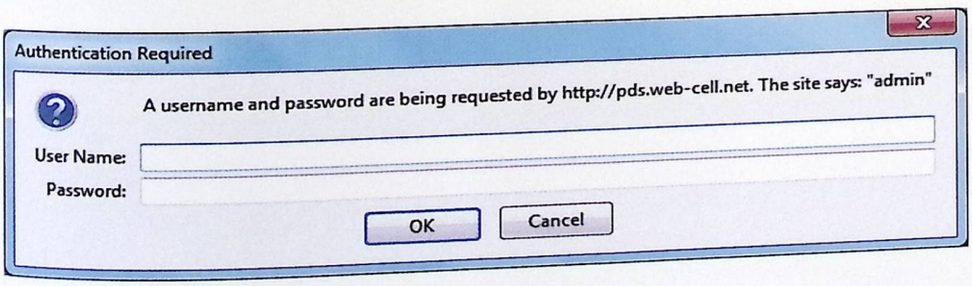


Figure 5-28: Password protected folder window

Chapter Six

System Implementation and Testing

Chapter Six

System Implementation and Testing

6.1 Overview

This chapter focuses on the methods and procedures used to implement, test and examine the system.

6.2 Testing Procedures

System testing is an important step in system implementation. It measures the effectiveness of the system before introducing it to the users.

The testing includes three main steps:

1. PDD testing.
2. Server testing.
3. The whole system testing.

6.2.1 PDD testing

PDD testing is divided into three stages:

1. PIC and LCD testing.
2. Keypad testing.
3. Ethernet testing.

PIC and LCD testing

This is accomplished using the following:

1. The sample code that comes with the CCS kit.
2. PIC C Compiler: this program is used to compile the sample code and to produce Hex files.
3. CCSLOAD: this program is used to write hex files to the kit.

The sample code that comes with the kit runs successfully. The LCD was working, some text viewed on it. That means that the PIC is working successfully.

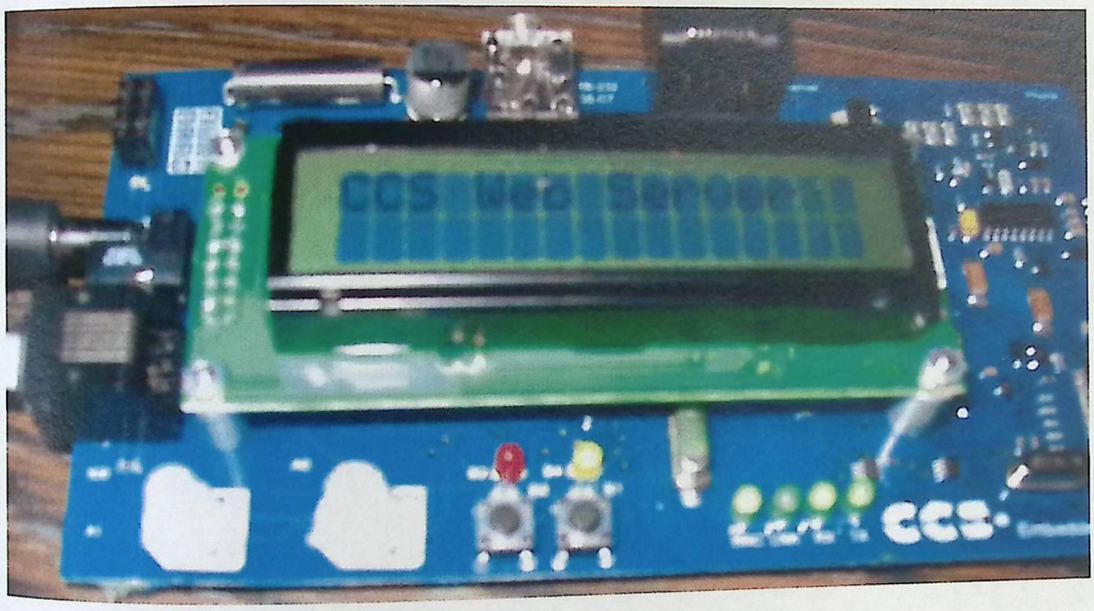


Figure 7-1: Initial kit testing

Keypad testing

This test aims to test the keypad and the interfacing circuit with the kit. Voltmeter was used to test the outputs of the interfacing circuit. A program that reads the keypad was written, so all buttons was tested.

Ethernet testing

The network card was tested using a sample program that discovers a network and get IP information. The IP, Gateway and subnet mask was viewed on the LCD to insure that the test was accomplished successfully.

After the kit has an IP, it was pinged from a PC to insure that the kit is connected.

Figure 7-2: Ethernet connection testing

The figure in the following page shows the PDD after it has been enclosed in a box.

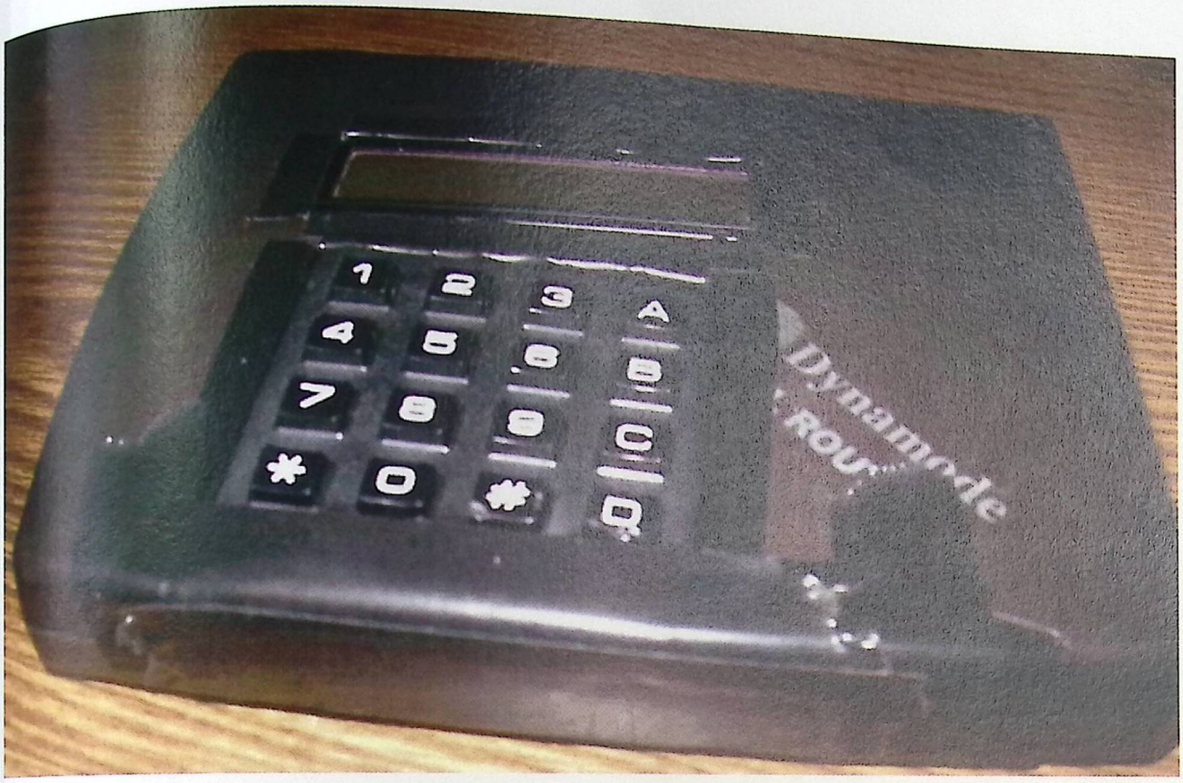


Figure 7-3: PDD

7.2.2 Server Testing

The server software was tested locally. All the software functions was tested and they was working successfully.

7.2.3 The whole system testing

PDS was tested to insure from the compatibility between PDS components. A testing search about some name was done, and the results was displayed.

Chapter Seven

Conclusions and Future work

Chapter Seven

Conclusion and Future Work

7.1 Conclusions:

Project conclusions:

1. The use of a development kit to implement the PDD is better for the following reasons:
   - Easier for programming by using the CCS code libraries.
   - Simpler to interface with external hardware, such as keypad.
   - Less expensive than other options.
   - More reliable.
2. The use of Ethernet connection:
   - Faster.
   - Simple to implement.
   - More reliable.
3. Providing an administration control panel in the server side, make it easier and more reliable to manage the PDS.
4. Dividing the work between team members, make it more comfortable and faster.

7.2 Future work

It is recommended to add the following ideas and features on this project:

1. Replace the traditional keypad and screen with a touch screen to improve the interface with users.
2. To provide a wireless internet connection instead of wired connection.
3. Add the ability to exchange messages between users and system administrators.
4. Enhance the project to be able to send advertisement to customers. That have an economic benefit for the provider company.

References

1. *Embedded Internet Development Kit. Retrieved March 20,2010 from Customer Computer Services(CSS) website:* http://www.ccsinfo.com/product_info.php?products_id=inetkit
2. *Free soft: IP Protocol Overview.* Retrieved March 8, 2010 from http://www.freesoft.org/CIE/Course/Section3/3.htm
3. *Free soft: TCP Protocol Overview* . Retrieved March 8, 2010 from http://www.freesoft.org/CIE/Course/Section3/3.htm
4. *Web developers notes: HTTP protocol- What is HTTP?*. Retrieved March 12, 2010 from http://www.webdevelopersnotes.com/basics/http_protocol.php3
5. *C Compiler Reference Manual.* 4th ed. CCS Inc, 2010.
6. *Development kit for PIC MCU Exercise Book.* CCS Inc, 2010.
7. Nigel Gardner. *PICmicro MCU C.* Bluebird Electronics 2002
8. Jeremy Bentham. *TCP/IP Lean.* 2nd ed. CMP Books.
9. *Resources for DHCP*. Retrieved October 17,2010 http://www.dhcp.org/
10. *Encoders.* Retrieved October 5,2010 http://mechatronics.mech.northwestern.edu/design_ref/sensors/encoders.html

Appendices

FAIRCHILD SEMICONDUCTOR™

October 1987
Revised January 1999

MM74C922 • MM74C923
16-Key Encoder • 20-Key Encoder

General Description

The MM74C922 and MM74C923 CMOS key encoders provide all the necessary logic to fully encode an array of SPST switches. The keyboard scan can be implemented by either an external clock or external capacitor. These encoders also have on-chip pull-up devices which permit switches with up to 50 kΩ on resistance to be used. No diodes in the switch array are needed to eliminate ghost switches. The internal debounce circuit needs only a single external capacitor and can be defeated by omitting the capacitor. A Data Available output goes to a high level when a valid keyboard entry has been made. The Data Available output returns to a low level when the entered key is released, even if another key is depressed. The Data Available will return high to indicate acceptance of the new key after a normal debounce period; this two-key roll-over is provided between any two switches.

An internal register remembers the last key pressed even after the key is released. The 3-STATE outputs provide for easy expansion and bus operation and are LPTTL compatible.

Features

- 50 kΩ maximum switch on resistance
- On or off chip clock
- On-chip row pull-up devices
- 2 key roll-over
- Keybounce elimination with single capacitor
- Last key register at outputs
- 3-STATE output LPTTL compatible
- Wide supply range: 3V to 15V
- Low power consumption

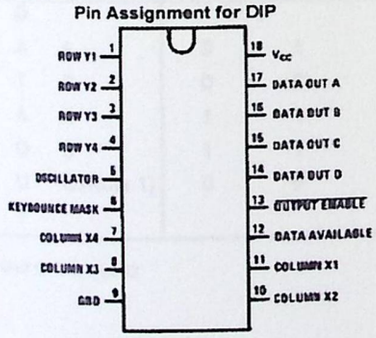
Ordering Code:

Order Number	Package Number	Package Description
MM74C922N	N18A	18-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
MM74C922WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300" Wide
MM74C923WM	M20B	20-Lead Small Outline Integrated Circuit (SOIC), JEDEC MS-013, 0.300" Wide
MM74C923N	N20A	20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide

Device also available in Tape and Reel. Specify by appending suffix letter "X" to the ordering code.

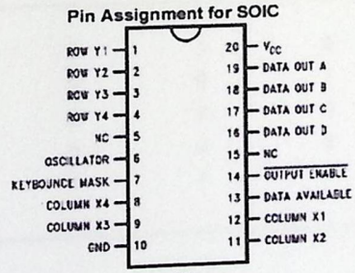
Connection Diagrams

Pin Assignment for DIP



Top View
MM94C922

Pin Assignment for SOIC

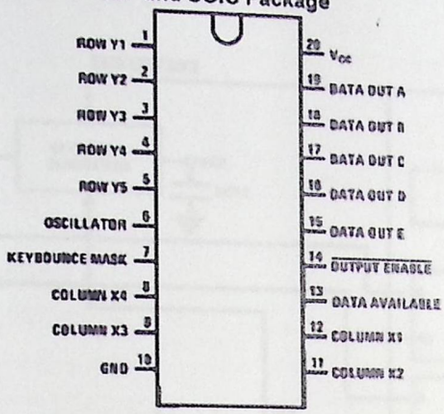


Top View
MM74C922

 DS006037.prf

Connection Diagrams (Continued)

Pin Assignment for
DIP and SOIC Package



Top View
MM74C923

Truth Tables

(Pins 0 through 11)

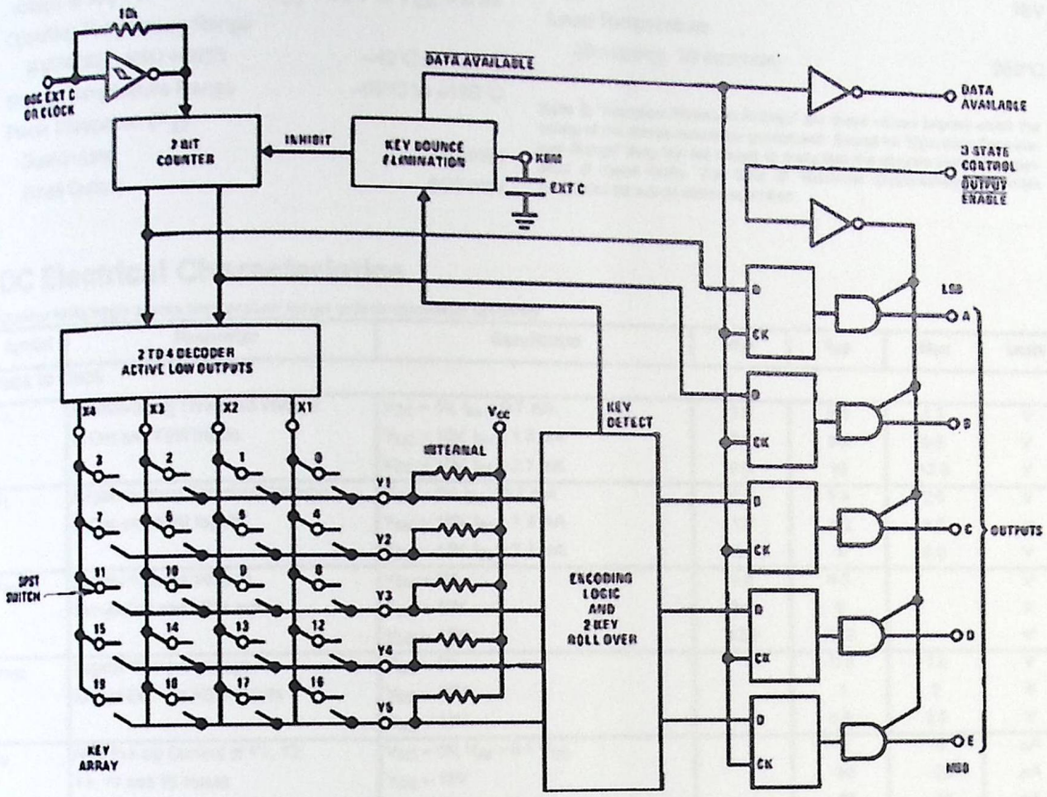
Switch Position	0 Y1,X1	1 Y1,X2	2 Y1,X3	3 Y1,X4	4 Y2,X1	5 Y2,X2	6 Y2,X3	7 Y2,X4	8 Y3,X1	9 Y3,X2	10 Y3,X3	11 Y3,X4
DATA A	0	1	0	1	0	1	0	1	0	1	0	1
DATA B	0	0	1	1	0	0	1	1	0	0	1	1
DATA C	0	0	0	0	1	1	1	1	0	0	0	0
DATA D	0	0	0	0	0	0	0	0	1	1	1	1
DATA E (Note 1)	0	0	0	0	0	0	0	0	0	0	0	0

(Pins 12 through 19)

Switch Position	12 Y4,X1	13 Y4,X2	14 Y4,X3	15 Y4,X4	16 Y5(Note 1), X1	17 Y5 (Note 1), X2	18 Y5 (Note 1), X3	19 Y5 (Note 1), X4
DATA A	0	1	0	1	0	1	0	1
DATA B	0	0	1	1	0	0	1	1
DATA C	1	1	1	1	0	0	0	0
DATA D	1	1	1	1	0	0	0	0
DATA E (Note 1)	0	0	0	0	1	1	1	1

Note 1: Omit for MM74C922

Block Diagram



Absolute Maximum Ratings (Note 2)

Voltage at Any Pin	V_{CC} − 0.3V to V_{CC} + 0.3V
Operating Temperature Range	
MM74C922, MM74C923	−40°C to +85°C
Storage Temperature Range	−65°C to +150°C
Power Dissipation (P_D)	
Dual-In-Line	700 mW
Small Outline	500 mW
Operating V_{CC} Range	3V to 15V
V_{CC}	18V
Lead Temperature	
(Soldering, 10 seconds)	260°C

Note 2: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. Except for "Operating Temperature Range" they are not meant to imply that the devices should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

DC Electrical Characteristics

Min/Max limits apply across temperature range unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Units
CMOS TO CMOS						
V_{T+}	Positive-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	3.0	3.6	4.3	V
		V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	6.0	6.8	8.6	V
		V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	9.0	10	12.9	V
V_{T-}	Negative-Going Threshold Voltage at Osc and KBM Inputs	V_{CC} = 5V, $I_{IN} \geq 0.7$ mA	0.7	1.4	2.0	V
		V_{CC} = 10V, $I_{IN} \geq 1.4$ mA	1.4	3.2	4.0	V
		V_{CC} = 15V, $I_{IN} \geq 2.1$ mA	2.1	5	6.0	V
$V_{IN(1)}$	Logical "1" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V	3.5	4.5		V
		V_{CC} = 10V	8.0	9		V
		V_{CC} = 15V	12.5	13.5		V
$V_{IN(0)}$	Logical "0" Input Voltage, Except Osc and KBM Inputs	V_{CC} = 5V		0.5	1.5	V
		V_{CC} = 10V		1	2	V
		V_{CC} = 15V		1.5	2.5	V
I_{IP}	Row Pull-Up Current at Y1, Y2, Y3, Y4 and Y5 Inputs	V_{CC} = 5V, V_{IN} = 0.1 V_{CC}		−2	−5	μA
		V_{CC} = 10V		−10	−20	μA
		V_{CC} = 15V		−22	−45	μA
$V_{OUT(1)}$	Logical "1" Output Voltage	V_{CC} = 5V, I_O = −10 μA	4.5			V
		V_{CC} = 10V, I_O = −10 μA	9			V
		V_{CC} = 15V, I_O = −10 μA	13.5			V
$V_{OUT(0)}$	Logical "0" Output Voltage	V_{CC} = 5V, I_O = 10 μA			0.5	V
		V_{CC} = 10V, I_O = 10 μA			1	V
		V_{CC} = 15V, I_O = 10 μA			1.5	V
R_{ON}	Column "ON" Resistance at X1, X2, X3 and X4 Outputs	V_{CC} = 5V, V_O = 0.5V		500	1400	Ω
		V_{CC} = 10V, V_O = 1V		300	700	Ω
		V_{CC} = 15V, V_O = 1.5V		200	500	Ω
I_{CC}	Supply Current, Osc at 0V, (one Y low)	V_{CC} = 5V		0.55	1.1	mA
		V_{CC} = 10V		1.1	1.9	mA
		V_{CC} = 15V		1.7	2.6	mA
$I_{IN(1)}$	Logical "1" Input Current at Output Enable	V_{CC} = 15V, V_{IN} = 15V		0.005	1.0	μA
$I_{IN(0)}$	Logical "0" Input Current at Output Enable	V_{CC} = 15V, V_{IN} = 0V	−1.0	−0.005		μA
CMOS/LPTTL INTERFACE						
$V_{IN(1)}$	Except Osc and KBM Inputs	V_{CC} = 4.75V	V_{CC} − 1.5			V
$V_{IN(0)}$	Except Osc and KBM Inputs	V_{CC} = 4.75V			0.8	V
$V_{OUT(1)}$	Logical "1" Output Voltage	I_O = −360 μA, V_{CC} = 4.75V	2.4			V
$V_{OUT(2)}$	Logical "0" Output Voltage	I_O = −360 μA, V_{CC} = 4.75V			0.4	V

DC Electrical Characteristics (Continued)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
OUTPUT DRIVE (See Family Characteristics Data Sheet) (Short Circuit Current)						
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 5V, V_{OUT} = 0V, T_A = 25°C	−1.75	−3.3		mA
I_{SOURCE}	Output Source Current (P-Channel)	V_{CC} = 10V, V_{OUT} = 0V, T_A = 25°C	−8	−15		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 5V, V_{OUT} = V_{CC}, T_A = 25°C	1.75	3.6		mA
I_{SINK}	Output Sink Current (N-Channel)	V_{CC} = 10V, V_{OUT} = V_{CC}, T_A = 25°C	8	16		mA

AC Electrical Characteristics (Note 3)

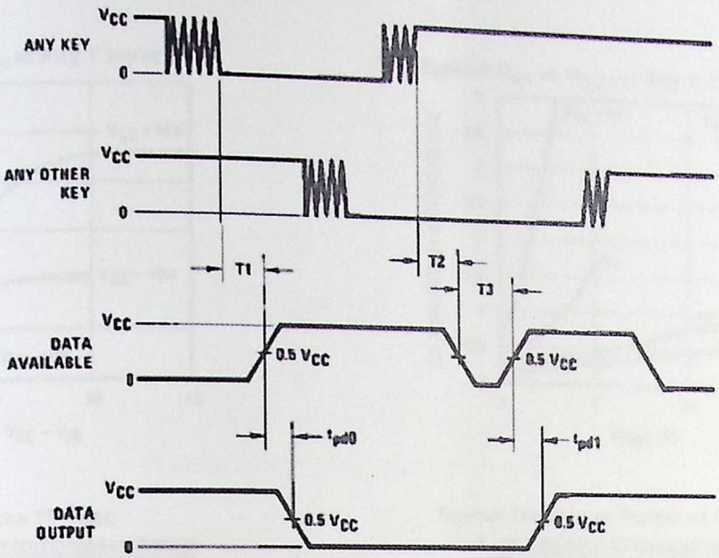
T_A = 25°C, C_L = 50 pF, unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{pd0}, t_{pd1}	Propagation Delay Time to Logical "0" or Logical "1" from D A	C_L = 50 pF (Figure 1)				
		V_{CC} = 5V		60	150	ns
		V_{CC} = 10V		35	80	ns
		V_{CC} = 15V		25	60	ns
t_{0H}, t_{1H}	Propagation Delay Time from Logical "0" or Logical "1" into High Impedance State	R_L = 10k, C_L = 10 pF (Figure 2)				
		V_{CC} = 5V, R_L = 10k		80	200	ns
		V_{CC} = 10V, C_L = 10 pF		65	150	ns
		V_{CC} = 15V		50	110	ns
t_{H0}, t_{H1}	Propagation Delay Time from High Impedance State to a Logical "0" or Logical "1"	R_L = 10k, C_L = 50 pF (Figure 2)				
		V_{CC} = 5V, R_L = 10k		100	250	ns
		V_{CC} = 10V, C_L = 50 pF		55	125	ns
		V_{CC} = 15V		40	90	ns
C_{IN}	Input Capacitance	Any Input (Note 4)		5	7.5	pF
C_{OUT}	3-STATE Output Capacitance	Any Output (Note 4)		10		pF

Note 3: AC Parameters are guaranteed by DC correlated testing.

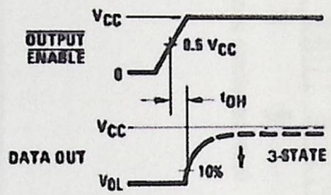
Note 4: Capacitance is guaranteed by periodic testing.

Switching Time Waveforms



T1 = T2 = RC, T3 = 0.7 RC, where R = 10k and C is external capacitor at KBM input.

FIGURE 1.



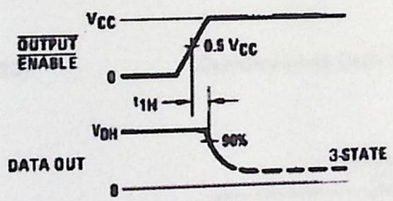
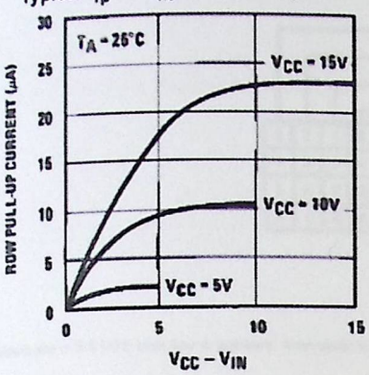


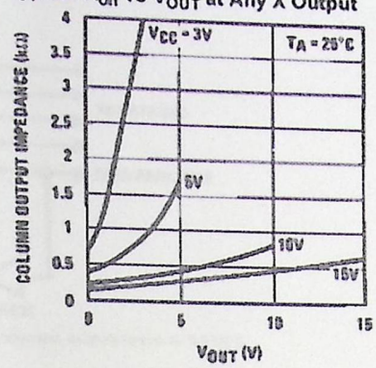
FIGURE 2.

Typical Performance Characteristics

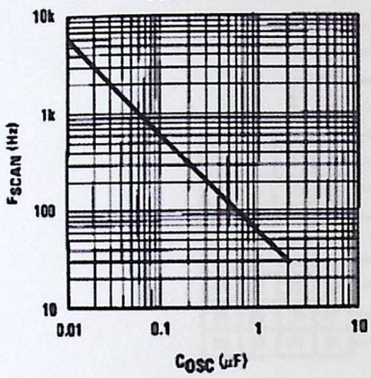
Typical I_{rp} vs V_{IN} at Any Y Input



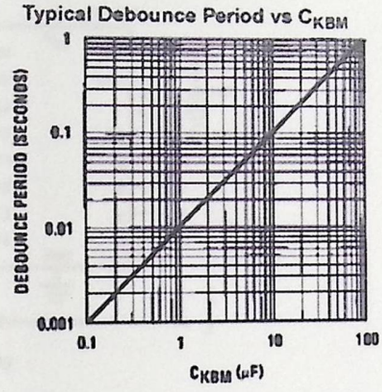
Typical R_{on} vs V_{OUT} at Any X Output



Typical F_{SCAN} vs C_{OSC}

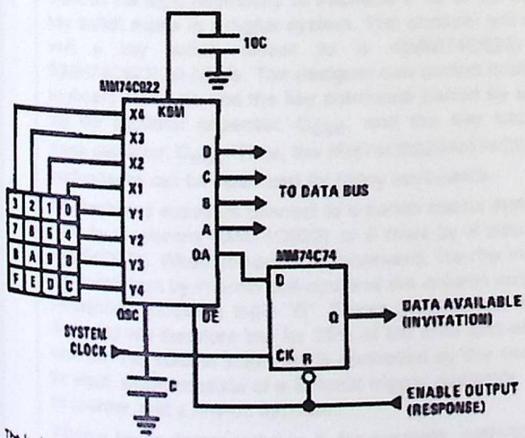


Typical Debounce Period vs C_{KBM}



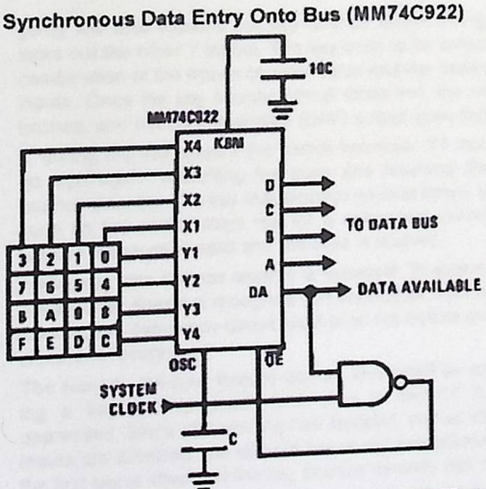
Typical Applications

Synchronous Handshake (MM74C922)



The keyboard may be synchronously scanned by omitting the capacitor at osc. and driving osc. directly if the system clock rate is lower than 10 kHz.

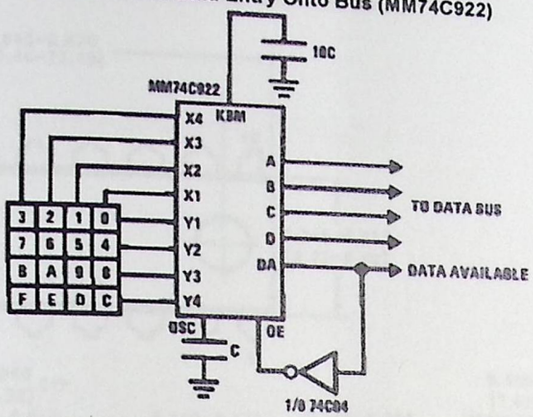
Synchronous Data Entry Onto Bus (MM74C922)



Outputs are enabled when valid entry is made and go into 3-STATE when key is released.

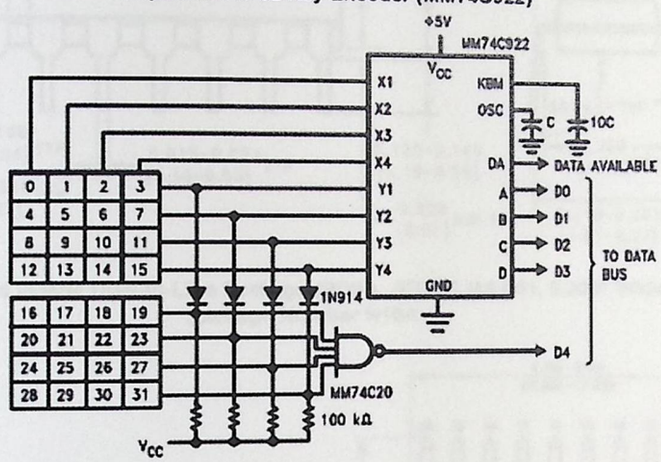
The keyboard may be synchronously scanned by omitting the capacitor at osc. and driving osc. directly if the system clock rate is lower than 10 kHz.

Asynchronous Data Entry Onto Bus (MM74C922)



Outputs are in 3-STATE until key is pressed, then data is placed on bus. When key is released, outputs return to 3-STATE.

Expansion to 32 Key Encoder (MM74C922)



Theory of Operation

The MM74C922/MM74C923 Keyboard Encoders implement all the logic necessary to interface a 16 or 20 SPST key switch matrix to a digital system. The encoder will convert a key switch closer to a 4(MM74C922) or 5(MM74C923) bit nibble. The designer can control both the keyboard scan rate and the key debounce period by altering the oscillator capacitor, C_{OSE}, and the key bounce mask capacitor, C_{MSK}. Thus, the MM74C922/MM74C923's performance can be optimized for many keyboards.

The keyboard encoders connect to a switch matrix that is 4 rows by 4 columns (MM74C922) or 5 rows by 4 columns (MM74C923). When no keys are depressed, the row inputs are pulled high by internal pull-ups and the column outputs sequentially output a logic "0". These outputs are open drain and are therefore low for 25% of the time and otherwise off. The column scan rate is controlled by the oscillator input, which consists of a Schmitt trigger oscillator, a 2-bit counter, and a 2–4-bit decoder.

When a key is depressed, key 0, for example, nothing will happen when the X1 input is off, since Y1 will remain high. When the X1 column is scanned, X1 goes low and Y1 will go low. This disables the counter and keeps X1 low. Y1 going low also initiates the key bounce circuit timing and locks out the other Y inputs. The key code to be output is a combination of the frozen counter value and the decoded Y inputs. Once the key bounce circuit times out, the data is latched, and the Data Available (DAV) output goes high.

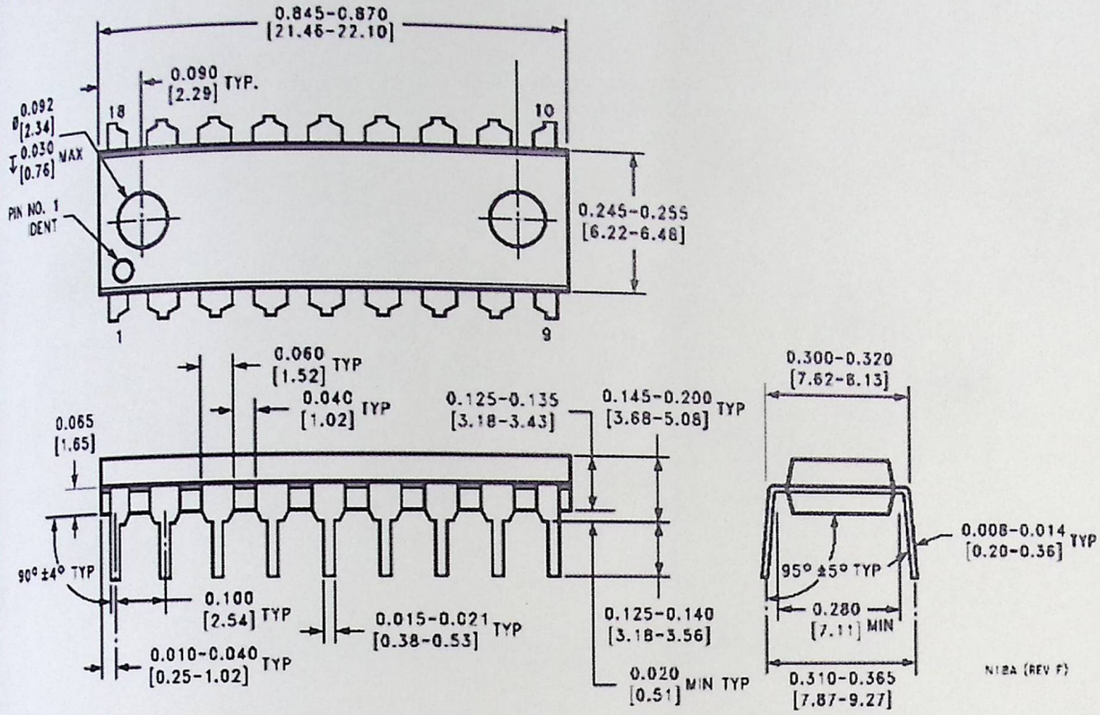
If, during the key closure the switch bounces, Y1 input will go high again, restarting the scan and resetting the key bounce circuitry. The key may bounce several times, but as soon as the switch stays low for a debounce period, the closure is assumed valid and the data is latched.

A key may also bounce when it is released. To ensure that the encoder does not recognize this bounce as another key closure, the debounce circuit must time out before another closure is recognized.

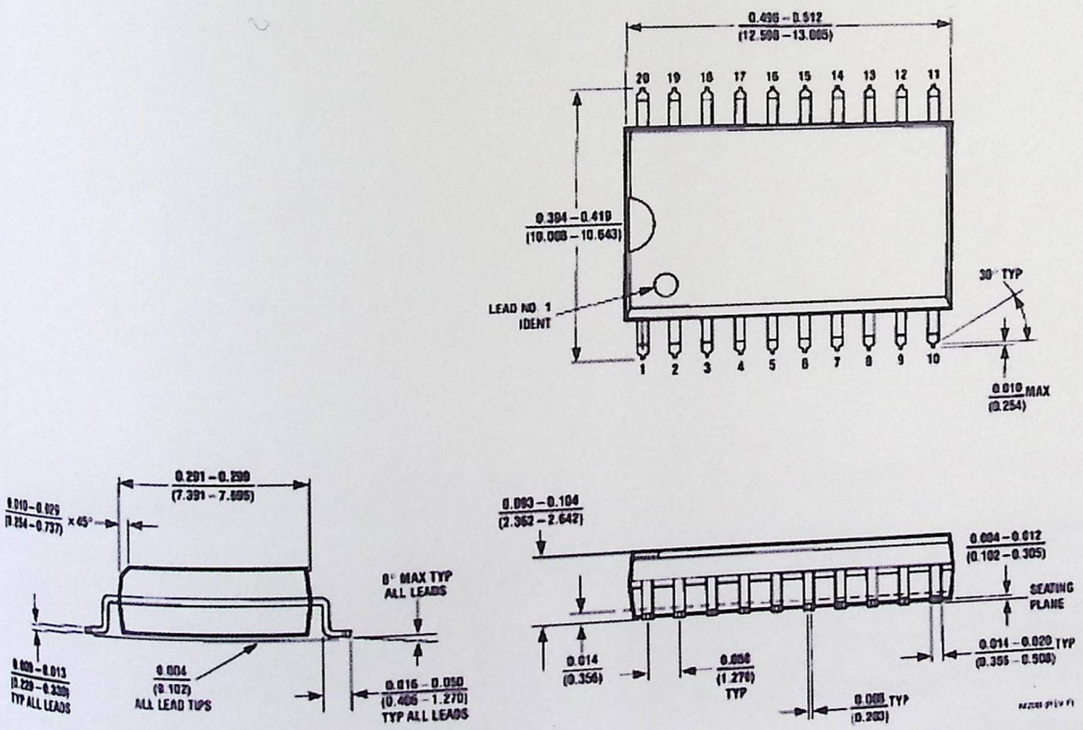
The two-key roll-over feature can be illustrated by assuming a key is depressed, and then a second key is depressed. Since all scanning has stopped, and all other Y inputs are disabled, the second key is not recognized until the first key is lifted and the key bounce circuitry has reset.

The output latches feed 3-STATE, which is enabled when the Output Enable ($\overline{OE}$) input is taken low.

Physical Dimensions inches (millimeters) unless otherwise noted

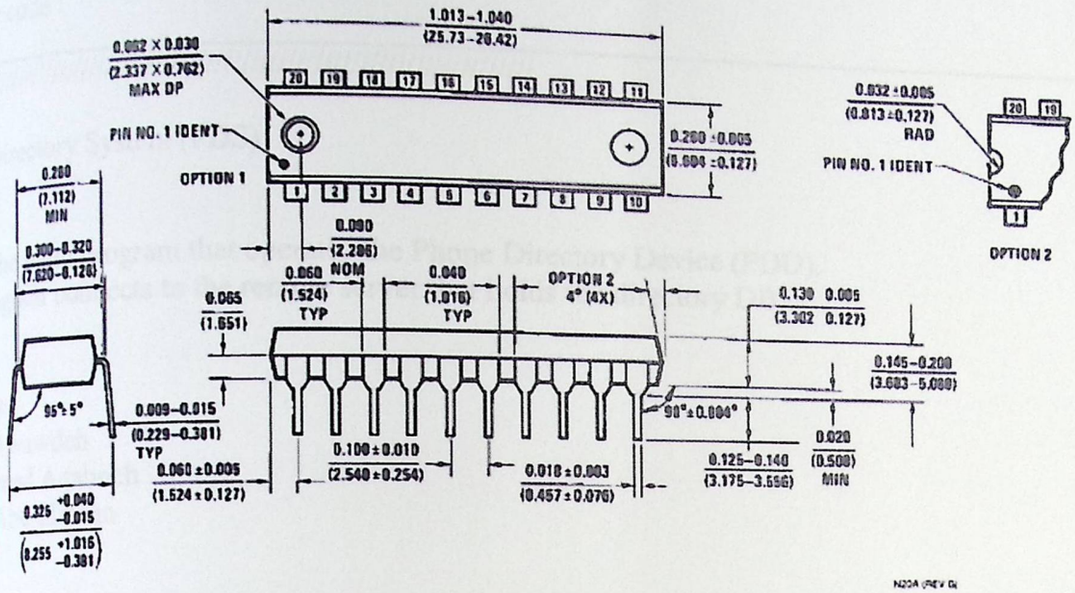


18-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
Package Number N18A



20-Lead Plastic Small Outline I.C. Package (M)
Package Number M20B

Physical Dimensions inches (millimeters) unless otherwise noted (Continued)



20-Lead Plastic Dual-In-Line Package (PDIP), JEDEC MS-001, 0.300" Wide
Package Number N20A

LIFE SUPPORT POLICY

FAIRCHILD'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF FAIRCHILD SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and (c) whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component in any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

www.fairchildsemi.com

Appendix B: PDD Code

pdd.c code :

```
//////////////////////////////////////////////////////////////////////
//
// Phone Directory System (PDS)
// pdd.c
//
// This is the main program that operates the Phone Directory Device (PDD).
// This program connects to the remote server that holds the directory DB.
//
//
// Coded By :
// Murad Awawdeh
// Mohammed Aqabneh
// Amjad Abu-Hassan
//
// 2010
//////////////////////////////////////////////////////////////////////


#define STACK_USE_ICMP  1
#define STACK_USE_ARP   1
#define STACK_USE_TCP   1
#define STACK_USE_DHCP  1
#define STACK_USE_UDP   1

#define SID "0000000001"
#define PASS "pds_pr123"

#define USE_DHCP        1 //When 1 the DHCP will assign IP and descover the
                          // network.
                          //when 0 the entered static IP info will be used.
#include "ccstcpip.h"

//keypad keys ::
//#define ANY_KEY   !input(PIN_B0)
//#define DOWN_ENTER !input(PIN_B0)
#define ANY_KEY    input(PIN_A4)
#define AB              !input(PIN_C5)&&!input(PIN_C2)&&!input(PIN_C1)&&!input(PIN_C0)
#define CD              !input(PIN_C5)&&!input(PIN_C2)&&!input(PIN_C1)&&input(PIN_C0)
#define EF              !input(PIN_C5)&&!input(PIN_C2)&&input(PIN_C1)&&!input(PIN_C0)
#define GH              !input(PIN_C5)&&!input(PIN_C2)&&input(PIN_C1)&&input(PIN_C0)
#define IJ              !input(PIN_C5)&&input(PIN_C2)&&!input(PIN_C1)&&!input(PIN_C0)
#define KL              !input(PIN_C5)&&input(PIN_C2)&&!input(PIN_C1)&&input(PIN_C0)
```

```
#define MN                !input(PIN_C5)&&input(PIN_C2)&&input(PIN_C1)&&!input(PIN_C0)
#define OP                !input(PIN_C5)&&input(PIN_C2)&&input(PIN_C1)&&input(PIN_C0)
#define QR                 input(PIN_C5)&&!input(PIN_C2)&&!input(PIN_C1)&&input(PIN_C0)
#define ST                 input(PIN_C5)&&!input(PIN_C2)&&!input(PIN_C1)&&!input(PIN_C0)
#define UV                 input(PIN_C5)&&!input(PIN_C2)&&!input(PIN_C1)&&input(PIN_C0)
#define WX                 input(PIN_C5)&&!input(PIN_C2)&&input(PIN_C1)&&!input(PIN_C0)
#define YZ_DIAL            input(PIN_C5)&&!input(PIN_C2)&&input(PIN_C1)&&input(PIN_C0)
#define UP_SPACE           input(PIN_C5)&&input(PIN_C2)&&!input(PIN_C1)&&!input(PIN_C0)
#define DOWN_ENTER input(PIN_C5)&&input(PIN_C2)&&input(PIN_C1)&&input(PIN_C0)
#define RST_BACK           input(PIN_C5)&&input(PIN_C2)&&input(PIN_C1)&&input(PIN_C0)
//=============================================================================
=
//=================================Global Variables=============================
enum {SPLASH,GET_IP,READY,GET_FIRST_NAME,GET_SECOND_NAME,GET_THIRD_NAME,
   GET_LAST_NAME,CONNECT,RESULT,SUSPENDED,NOT_ADDED} status=SPLASH;
long count1=0,count; // contains time between buttons pressing
int key=2; // determin the function of key if multifunction
int last_key=0;// last key pressed
int i; // contains no. of word's letters
IP_ADDR server;//IP address of the remote web server
#define PORT   3128// the remote port
TCP_SOCKET socket=INVALID_SOCKET;// the connection socket
// == the name of the person looking for his phone no. ==
char first_name[20];
char second_name[20];
char third_name[20];
char last_name[20];
//==========
char results[100][20];
int num_of_res=0;
//=============================================Functions Prototypes=============
void key_delay(void);
int8 HTTPConnectedTask();
void HTTPTask();
void ServerAddrInit(void);
void print_key(int key_no,char x,char y);
void get_word(char ar[]);
void splash();
void marquee(int ,char *);

//========== Main Function ====================================================
void main(void)
{

  // initializations ..
  MACAddrInit();
```

```
 ServerAddrInit();
 init_user_io();
 lcd_init();
 enable_interrupts(INT_TIMER1);
 enable_interrupts(GLOBAL);
 StackInit();
//=============================================================================
=

while(1) // the main program while
{
 switch(status)
 {
  case SPLASH:
   printf(lcd_putc,"\f");
   splash();


   while(!ANY_KEY)
   {
   }; // wait until any key is pressed
   key_delay();
   status=GET_FIRST_NAME;
  break;
  //=============================================================================
  case GET_IP:

   if(USE_DHCP)
   {
    printf(lcd_putc,"\fGetting IP .. :");
    while(!DHCPIsBound())
    {

     if (!MACIsLinked()) {
      printf(lcd_putc,"\nNo Ethernet          ");
     }
     else if (!DHCPIsBound()) {
      printf(lcd_putc,"\nDCHP Not Bound       ");
     }

     StackTask();
    }
   }
   else
    IPAddrInit(); // Statically assigns IP,GW,SM
```

```
   printf(lcd_putc,"\fIP:%u.%u.%u.%u", MY_IP_BYTE1, MY_IP_BYTE2,
      MY_IP_BYTE3, MY_IP_BYTE4);
   printf(lcd_putc,"\nGW:%u.%u.%u.%u", MY_GATE_BYTE1, MY_GATE_BYTE2,
      MY_GATE_BYTE3, MY_GATE_BYTE4);

   while(!ANY_KEY); // wait until any key is pressed
   key_delay();
   status=READY;
 break;
 //=============================================================================
 case READY:
   printf(lcd_putc,"\fReady");
   for(k=0;k<=255;k++)
   {
   printf(lcd_putc,"\f%d : %c ",k,k);
   delay_ms(1000);
   }

   while(!ANY_KEY); // wait until any key is pressed
   key_delay();
   status=CONNECT;
 break;
 //=============================================================================
 case GET_FIRST_NAME:
   printf(lcd_putc,"\fFirst Name :\n");
   get_word(first_name);
   status=GET_SECOND_NAME;
 break;
 //=============================================================================
 case GET_SECOND_NAME:
    printf(lcd_putc,"\fSecond Name :\n");
    get_word(second_name);
    status=GET_THIRD_NAME;
 break;
 //=============================================================================
 case GET_THIRD_NAME:
   printf(lcd_putc,"\fThird Name :\n");
   get_word(third_name);
   status=GET_LAST_NAME;
 break;
 //=============================================================================
 case GET_LAST_NAME:
   printf(lcd_putc,"\fFamily Name :\n");
   get_word(last_name);
   status=CONNECT;
  //printf(lcd_putc,"\f");
```

```
    break;
    //=============================================================================
    case CONNECT:

      StackTask();

      HTTPTask();


    break;
    //=============================================================================
    case RESULT:
      for(j=0;j<num_of_res;j++)
      {
      printf(lcd_putc,"\f#%d\nNO:%s",j+1,results[j]);
      while(!ANY_KEY);
      key_delay();
      }

    break;
    //=============================================================================
    case SUSPENDED:
      printf(lcd_putc,"\fThis PDD is\nSuspended");
      while(!ANY_KEY);
    break;
    //=============================================================================
    case NOT_ADDED:
      printf(lcd_putc,"\fPDD isn't added\n to the system");
      while(!ANY_KEY);
    break;
  }// end switch status
}// end of the main while



}// End of Main function

//================== Functions ================================================

// This function provides delay between buttons pressed
void key_delay(void)
{
  delay_ms(200);
}
//=============================================================================
=
```

```
//this function is called by MyTCPTask() when the specified socket is connected
//to the web server.
int8 HTTPConnectedTask() {
  char c;
  char str[255];
  int x,y;

  if (TCPIsGetReady(socket)) {
    if(TCPGet(socket, &c))
      if(c=='N')
      {
        status=NOT_ADDED;
      }
      else if(c=='S')
      {
        status=SUSPENDED;
      }
      else if(c=='O')
      {
      y=-1;
      x=0;
       while (TCPGet(socket, &c)) {
          if(c=='X')
          {
            if(y!=-1)
              results[y][x]='\0';
            y++;
            x=0;
          }
          else
          {
            results[y][x++]=c;
          }
       }
       results[y][x]='\0';
       num_of_res=y+1;
       status=RESULT;
      }
  }// end of tcp is get ready

  if (ANY_KEY && DOWN_ENTER && TCPIsPutReady(socket)) {
    printf(lcd_putc,"\fSending Request .. ");
    sprintf(str,"GET GET http://pds.web-cell.net/search.php?pass=%s&sid=%s &fname=%s&sname=%s
&tname=%s&lname=%s\n nHost: pds.web-cell.net ", PASS,SID, first_name, second_name, third_name,
last_name);
```

```
   TCPPutArray(socket,str,strlen(str));
   TCPFlush(socket);

  }
  /*
  if(ANY_KEY && RST_BACK)
      {
        status=GET_IP;
        key_delay();
        return(TRUE);
      }
*/
  return(FALSE);
}
//=============================================================================
=

//This function connects to the remote server
void HTTPTask() {
  static TICKTYPE lastTick;

  static enum {
    MYTCP_STATE_NEW=0, MYTCP_STATE_ARP_REQ=1, MYTCP_STATE_ARP_WAIT=2,
    MYTCP_STATE_CONNECT=3, MYTCP_STATE_CONNECT_WAIT=4,
    MYTCP_STATE_CONNECTED=5, MYTCP_STATE_DISCONNECT=6,
    MYTCP_STATE_FORCE_DISCONNECT=7
  } state=0;
  static NODE_INFO remote;
  TICKTYPE currTick;
  int8 dis;

  currTick=TickGet();

  switch (state) {
    case MYTCP_STATE_NEW:
      memcpy(&remote.IPAddr, &server, sizeof(IP_ADDR));
      printf(lcd_putc,"\nARP REQUEST     ");
      state=MYTCP_STATE_ARP_REQ;

    case MYTCP_STATE_ARP_REQ:
      if (ARPIsTxReady()) {
        ARPResolve(&remote.IPAddr);
        lastTick=currTick;
        state=MYTCP_STATE_ARP_WAIT;
      }
```

```
    break;

  case MYTCP_STATE_ARP_WAIT:
    if (ARPIsResolved(&remote.IPAddr, &remote.MACAddr)) {
      state=MYTCP_STATE_CONNECT;
      printf(lcd_putc,"\nCONNECTING     ");
    }
    else if (TickGetDiff(currTick, lastTick) > (TICKS_PER_SECOND * 2)) {
      state=MYTCP_STATE_ARP_REQ;
    }
    break;

  case MYTCP_STATE_CONNECT:
    socket=TCPConnect(&remote, PORT);
    if (socket!=INVALID_SOCKET) {
      lastTick=TickGet();
      state=MYTCP_STATE_CONNECT_WAIT;
    }
    else {
      printf(lcd_putc,"\nSOCKET ERROR    ");
    }
    break;

  case MYTCP_STATE_CONNECT_WAIT:
    if (TCPIsConnected(socket)) {
      state=MYTCP_STATE_CONNECTED;
      printf(lcd_putc,"\nCONNECTED!      ");
    }
    else if (TickGetDiff(currTick, lastTick) > (TICKS_PER_SECOND * 10)) {
      state=MYTCP_STATE_FORCE_DISCONNECT;


    }
    break;

  case MYTCP_STATE_CONNECTED:
    if (TCPIsConnected(socket)) {

      dis=HTTPConnectedTask();
      if (dis) {
        state=MYTCP_STATE_DISCONNECT;
        lastTick=currTick;
      }

    }
    else {
```

```
      printf(lcd_putc,"\nDISCONNECTED   ");
      state=MYTCP_STATE_CONNECT;
    }
    break;

  case MYTCP_STATE_DISCONNECT:
    printf(lcd_putc,"\nDISCONNECTING   ");
    if (TCPIsPutReady(socket)) {
      state=MYTCP_STATE_FORCE_DISCONNECT;
    }
    else if (TickGetDiff(currTick, lastTick) > (TICKS_PER_SECOND * 10)) {
      state=MYTCP_STATE_FORCE_DISCONNECT;
    }
    break;

  case MYTCP_STATE_FORCE_DISCONNECT:
    TCPDisconnect(socket);
    state=MYTCP_STATE_CONNECT;

    break;
  }
}
//=================================================================================
=

//This function initialize the ip address of the remote web server
void ServerAddrInit(void) {
  // 10.2.1.8
  server.v[0]=10;
  server.v[1]=2;
  server.v[2]=1;
  server.v[3]=8;
}
//=================================================================================
=

//This function determins which letter to be printed and print it, since each
//button prints two letters
void print_key(int key_no,char x,char y)
{
  if(last_key==key_no&&count<15&&key==1)
  {
        printf(lcd_putc,"\b%c",y);
        key=2;

  }
```

```
    printf(lcd_putc,"\nDISCONNECTED    ");
    state=MYTCP_STATE_CONNECT;
  }
  break;

case MYTCP_STATE_DISCONNECT:
  printf(lcd_putc,"\nDISCONNECTING   ");
  if(!TCPIsPutReady(socket)) {
    state=MYTCP_STATE_FORCE_DISCONNECT;
  }
  else if(TickGetDiff(currTick, lastTick) > (TICKS_PER_SECOND * 10)) {
    state=MYTCP_STATE_FORCE_DISCONNECT;
  }
  break;

case MYTCP_STATE_FORCE_DISCONNECT:
  TCPDisconnect(socket);
  state=MYTCP_STATE_CONNECT;

  break;
}
```

```
  if(AB&& i<15)

    print_key(1,'A','B');
    last_key=1;

  else if(CD&& i<15)

    print_key(2,'C','D');
    last_key=2;

  else if(EF&& i<15)
```

```
  else if(last_key==key_no&&count<15&&key==2)
  {
          printf(lcd_putc,"\b%c",x);
          key=1;


  }
  else
  {
          printf(lcd_putc,"%c",x);
          key=1;
          i++;
  }
}
//=============================================================================
=

//This function reads a word from the the LCD buffer and stores it in array of
//characters
void get_word(char ar[])
{
int j;
i=0;
  while(1)
  {
    if(ANY_KEY)
    {

        count=count1;
        setup_timer_1(T1_INTERNAL|T1_DIV_BY_8);
        set_timer1(0);
        count1=0;

        //i++;

      if(AB && i<15)
      {
        print_key(1,'A','B');
        last_key=1;
      }
      else if(CD&& i<15)
      {
        print_key(2,'C','D');
        last_key=2;
      }
      else if(EF&& i<15)
      {
```

```
    print_key(3,'E','F');
    last_key=3;
  }
  else if(GH&& i<15)
  {
    print_key(4,'G','H');
    last_key=4;
  }
  else if(IJ&& i<15)
  {
    print_key(5,'I','J');
    last_key=5;
  }
  else if(KL&& i<15)
  {
    print_key(6,'K','L');
    last_key=6;
  }
  else if(MN&& i<15)
  {
    print_key(7,'M','N');
    last_key=7;
  }
  else if(OP&& i<15)
  {
    print_key(8,'O','P');
    last_key=8;
  }
  else if(QR&& i<15)
  {
    print_key(9,'Q','R');
    last_key=9;
  }
  else if(ST&& i<15)
  {
    print_key(10,'S','T');
    last_key=10;
  }
  else if(UV&& i<15)
  {
    print_key(11,'U','V');
    last_key=11;
  }
  else if(WX&& i<15)
  {
    print_key(12,'W','X');
```

```
      last_key=12;
    }
    else if(YZ_DIAL&& i<15)
    {
      print_key(13,'Y','Z');
      last_key=13;
    }
    else if(UP_SPACE&& i<15)
    {
      printf(lcd_putc," ");
      last_key=14;
    }
    else if(DOWN_ENTER)
    {
      break;
    }
    else if((RST_BACK)&&i>0)
    {

      i--;

      last_key=16;
      printf(lcd_putc,"\b \b");
    }


    key_delay();
  }// end if any key
}// end while 1

for(j=1;j<=i;j++)
  ar[j-1]=lcd_getc(j,2);

ar[j-1]='\0';

key_delay();
}
//=============================================================================
=
// splash screen displayed on PDD start
void splash()
{
 lcd_gotoxy(1,1);

printf(lcd_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",255,255,255,255,255,255,255,255,25
5,255,255,255,255,255,255,255);
```

```
      lcd_gotoxy(1,2);

printf(lcd_putc,"%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c%c",255,255,255,255,255,255,255,255,25
5,255,255,255,255,255,255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);

printf(lcd_putc,"%c%c%c%c%c%c%cDI%c%c%c%c%c%c%c",255,255,255,255,255,255,255,255,255,25
5,255,255,255,255);
      lcd_gotoxy(1,2);

printf(lcd_putc,"%c%c%c%c%c%c%cST%c%c%c%c%c%c%c",255,255,255,255,255,255,255,255,255,2
55,255,255,255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);
      printf(lcd_putc,"%c%c%c%c%c%c%c
DIR%c%c%c%c%c%c%c",255,255,255,255,255,255,255,255,255,255,255,255,255);
      lcd_gotoxy(1,2);

printf(lcd_putc,"%c%c%c%c%c%c%cYSTE%c%c%c%c%c%c%c",255,255,255,255,255,255,255,255,255,255,
255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);
      printf(lcd_putc,"%c%c%c%c%c%cE
DIRE%c%c%c%c%c%c",255,255,255,255,255,255,255,255,255,255,255,255);
      lcd_gotoxy(1,2);

printf(lcd_putc,"%c%c%c%c%c%cSYSTEM%c%c%c%c%c%c",255,255,255,255,255,255,255,255,255,255,255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);
      printf(lcd_putc,"%c%c%c%c%cNE DIREC%c%c%c%c%c",255,255,255,255,255,255,255,255,255,255);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"%c%c%c%c%c SYSTEM %c%c%c%c%c",255,255,255,255,255,255,255,255,255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);
      printf(lcd_putc,"%c%c%c%cONE DIRECT%c%c%c%c",255,255,255,255,255,255,255,255);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"%c%c%c%c  SYSTEM  %c%c%c%c%c",255,255,255,255,255,255,255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);
```

```
      printf(lcd_putc,"%c%cHONE DIRECTO%c%c",255,255,255,255);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"%c%c   SYSTEM   %c%c",255,255,255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);
      printf(lcd_putc,"%cPHONE DIRECTOR%c",255,255);
      lcd_gotoxy(1,2);
      printf(lcd_putc,"%c    SYSTEM    %c",255,255);
      delay_ms(200);

      lcd_gotoxy(1,1);
      printf(lcd_putc," PHONE DIRECTORY");
      lcd_gotoxy(1,2);
      printf(lcd_putc,"     SYSTEM     ");
      delay_ms(200);
}
//=============================================================================
=
//
void marquee(int l,char *str)
{

}
//=============================================================================
=
#INT_TIMER1 //This indicates the interrupt service routine

//This is the interrupt service routine, it increase a counter when timer1 is
//overflow
void isr(void)
{
  count1++;
}
```

ccstcpip.h (written by CSS)

```
////////////////////////////////////////////////////////////////////////
//
// ccstcpip.h - Common code shared among all Embedded Internet/Embedded
// Ethernet tutorial book examples.
//
// If you are using a CCS Embedded Ethernet Board (labeled PICENS, which
// has an MCP ENC28J60) then define STACK_USE_CCS_PICENS to TRUE.
//
// If you are using a CCS Embedded Internet Board (labeled PICNET, which
// has a Realtek RTL8019AS and a 56K Modem) then define STACK_USE_CCS_PICNET
// to TRUE.
//
////////////////////////////////////////////////////////////////////////
//
// 10/25/06
// - Added STACK_USE_CCS_PICEEC
// - ExampleUDPPacket[] UDP header length fixed
//
////////////////////////////////////////////////////////////////////////

#define STACK_USE_CCS_PICENS   0 //18f4620 + enc28j60
#define STACK_USE_CCS_PICNET   1 //18f6722 + realtek
#define STACK_USE_CCS_PICEEC   0 //18f97j60

#if STACK_USE_CCS_PICENS
 #define STACK_USE_MCPENC 1
#endif

#if STACK_USE_CCS_PICEEC
 #define STACK_USE_MCPINC 1
#endif

#if STACK_USE_CCS_PICENS
 #include <18F4620.h>
 #use delay(clock=40000000)
 #fuses H4, NOWDT, NOLVP, NODEBUG
#elif STACK_USE_CCS_PICNET
 #include <18F6722.h>
 #use delay(clock=40000000)
 #fuses H4, NOWDT, NOLVP, NODEBUG
#elif STACK_USE_CCS_PICEEC
 #include <18F67J60.h>
 //#use delay(clock=41666667)
 //#fuses NOWDT, NODEBUG, H4_SW, NOIESO, NOFCMEN, PRIMARY
 #use delay(clock=25M)
```

```
 #fuses NOWDT, NODEBUG, HS, NOIESO, NOFCMEN, PRIMARY, ETHLED
#elif STACK_USE_CCS_PICNET_OLD
 #include <18F6720.h>
 #use delay(clock=20000000)
 #fuses HS, NOWDT, NOLVP, NODEBUG
 #define STACK_USE_CCS_PICNET   TRUE
#else
 #error You need to define your custom hardware
#endif

#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

#include "tcpip/stacktsk.c"    //include Microchip TCP/IP Stack

#if STACK_USE_CCS_PICENS
 #include "tcpip/mlcd.c"
 #define BUTTON1_PRESSED()  (!input(PIN_A4))
 #define USER_LED1    PIN_A5
 #define USER_LED2    PIN_B4
 #define USER_LED3    PIN_B5
 #define LED_ON       output_low
 #define LED_OFF      output_high
 #define STANDARD_ADC_STRING  "AN0"
 #define STANDARD_ADC_CHANNEL 0
 void init_user_io(void) {
  setup_adc(ADC_CLOCK_INTERNAL);
  setup_adc_ports(AN0);
  *0xF92=(*0xF92 & 0xDF) | 0x11;   //a5 output, a4 and a0 input
  *0xF93=*0xF93 & 0xCF;   //b4 and b5 output
  LED_OFF(USER_LED1);
  LED_OFF(USER_LED2);
  LED_OFF(USER_LED3);
  set_adc_channel(0);
 }
#elif STACK_USE_CCS_PICEEC
 #include "tcpip/elcd.c"
 #define BUTTON1_PRESSED()  (!input(PIN_A4))
 #define USER_LED1    PIN_B3
 #define USER_LED2    PIN_B4
 #define USER_LED3    PIN_B5
 #define LED_ON       output_low
 #define LED_OFF      output_high
 #define STANDARD_ADC_STRING  "AN2"
 #define STANDARD_ADC_CHANNEL 2
 void init_user_io(void) {
  setup_adc(ADC_CLOCK_INTERNAL);
```

```
  setup_adc_ports(AN0_TO_AN2);
  set_adc_channel(2);
  *0xF92=*0xF92 & 0xFC;   //a0 and a1 output
  *0xF93=*0xF93 & 0xC7;   //b3, b4 and b5 output
  LED_OFF(USER_LED1);
  LED_OFF(USER_LED2);
  LED_OFF(USER_LED3);
  set_adc_channel(2);
 }
#else
 #include "tcpip/dlcd.c"
 #define BUTTON1_PRESSED()  (!input(PIN_B0))
 #define BUTTON2_PRESSED()  (!input(PIN_B1))
 #define USER_LED1    PIN_B2
 #define USER_LED2    PIN_B4
 #define LED_ON       output_low
 #define LED_OFF      output_high
 #define STANDARD_ADC_STRING  "AN0"
 #define STANDARD_ADC_CHANNEL 0
 void init_user_io(void) {
  setup_adc(ADC_CLOCK_INTERNAL );
  setup_adc_ports(ANALOG_AN0_TO_AN1);
  *0xF92=*0xF92 | 3;           //a0 and a1 input (for ADC)
  *0xF93=(*0xF93 & 0xEB) | 3;   //B0 and B1 input, B2 and B4 output
  LED_OFF(USER_LED1);
  LED_OFF(USER_LED2);
  set_adc_channel(0);
 }
#endif

void MACAddrInit(void) {
  MY_MAC_BYTE1=0;
  MY_MAC_BYTE2=2;
  MY_MAC_BYTE3=3;
  MY_MAC_BYTE4=4;
  MY_MAC_BYTE5=5;
  MY_MAC_BYTE6=6;
}

void IPAddrInit(void) {
  //IP address of this unit
  MY_IP_BYTE1=192;
  MY_IP_BYTE2=168;
  MY_IP_BYTE3=0;
  MY_IP_BYTE4=7;
```

```
  setup_adc_ports(AN0_TO_AN2);
  set_adc_channel(2);
  *0xF92=*0xF92 & 0xFC;   //a0 and a1 output
  *0xF93=*0xF93 & 0xC7;   //b3, b4 and b5 output
  LED_OFF(USER_LED1);
  LED_OFF(USER_LED2);
  LED_OFF(USER_LED3);
  set_adc_channel(2);
 }
#else
 #include "tcpip/dlcd.c"
 #define BUTTON1_PRESSED()  (!input(PIN_B0))
 #define BUTTON2_PRESSED()  (!input(PIN_B1))
 #define USER_LED1    PIN_B2
 #define USER_LED2    PIN_B4
 #define LED_ON       output_low
 #define LED_OFF      output_high
 #define STANDARD_ADC_STRING  "AN0"
 #define STANDARD_ADC_CHANNEL 0
 void init_user_io(void) {
  setup_adc(ADC_CLOCK_INTERNAL );
  setup_adc_ports(ANALOG_AN0_TO_AN1);
  *0xF92=*0xF92 | 3;           //a0 and a1 input (for ADC)
  *0xF93=(*0xF93 & 0xEB) | 3;   //B0 and B1 input, B2 and B4 output
  LED_OFF(USER_LED1);
  LED_OFF(USER_LED2);
  set_adc_channel(0);
 }
#endif

void MACAddrInit(void) {
  MY_MAC_BYTE1=0;
  MY_MAC_BYTE2=2;
  MY_MAC_BYTE3=3;
  MY_MAC_BYTE4=4;
  MY_MAC_BYTE5=5;
  MY_MAC_BYTE6=6;
}

void IPAddrInit(void) {
  //IP address of this unit
  MY_IP_BYTE1=192;
  MY_IP_BYTE2=168;
  MY_IP_BYTE3=0;
  MY_IP_BYTE4=7;
```

```
  //network gateway
  MY_GATE_BYTE1=192;
  MY_GATE_BYTE2=168;
  MY_GATE_BYTE3=0;
  MY_GATE_BYTE4=1;

  //subnet mask
  MY_MASK_BYTE1=255;
  MY_MASK_BYTE2=255;
  MY_MASK_BYTE3=255;
  MY_MASK_BYTE4=0;
}

char ExampleIPDatagram[] = {
  0x45, 0x00, 0x00, 0x20, 0x00, 0x00, 0x00, 0x00,
  0x64, 0x11, 0x2A, 0x9D, 0x0A, 0x0B, 0x0C, 0x0D,
  0x0A, 0x0B, 0x0C, 0x0E
};

char ExampleUDPPacket[] = {
  0x04, 0x00, 0x04, 0x01, 0x00, 0x08, 0x00, 0x00,
  0x01, 0x02, 0x03, 0x04
};
```

Appendix C: Server Code

index.php

```
<?php
echo "Restricted Access";
?>
```

search.php

```
<?php

include("includes/db.php");
include("includes/device.php");
include("includes/directory.php");
include("includes/log.php");

$pass=@$_GET['pass'];

if($pass=="pds_pr123")
{


   $sid=@$_GET['sid'];
   $dev=new Device($sid);
   $ch=$dev->Check();
   if($ch=="O")
   {
     echo $ch;

     $fname=@$_GET['fname'];
     $sname=@$_GET['sname'];
     $tname=@$_GET['tname'];
     $lname=@$_GET['lname'];

     $lookedfor=$fname." ".$sname." ".$tname." ".$lname;

     if($sname=="")
       $sname="%%";
     if($tname=="")
       $tname="%%";

     $query=new Ph_Directory($fname,$sname,$tname,$lname);
     $q=$query->Perform_Query();
     //echo mysql_num_rows($q);

     while ($r=mysql_fetch_array($q))
     {
       echo "X";
       echo $r['PhoneNo'];
     }

     $l=new Log($sid,$_SERVER['REMOTE_ADDR'],$lookedfor);
     $l->Add();
}
```

```
    else
      echo $ch;

}
else
{
  echo "Restricted Access";
}
?>
```

includes/admin.php

```
<?php

class Admin
{
  var $uname;
  var $name;
  var $email;
  var $upass;
  var $con;



  function Admin($uname="",$upass="",$name="",$email="")
  {
    $this->uname=$uname;
    $this->name=$name;
    $this->email=$email;
    $this->upass=$upass;

    $newdb=new db();
    $this->con=$newdb->connect();
  }


  function Create_Admin()
  {
    $q=@mysql_query("select UName from admins where UName='$this->uname'",$this->con);
    $t=@mysql_fetch_array($q);

      if($t)
        return "existuser";

    $q=@mysql_query("select UName from admins where Email='$this->email'",$this->con);
    $t=@mysql_fetch_array($q);

      if($t)
        return "existemail";

    $r=@mysql_query("insert into admins(UName,Name,Email,UPass) values('$this->uname','$this-
>name','$this->email',md5('$this->upass'))",$this->con);
      if($r)
        return "done";
      else
        return "error";
  }
```

```
 function Is_Admin()
 {
   $q=@mysql_query("select * from admins where UName='$this->uname' and UPass=md5('$this-
>upass')",$this->con);
   $r=@mysql_fetch_array($q);


     return $r;

 }

 function Show_Admin($aid)
 {
  $q=@mysql_query("select * from admins where AID=$aid",$this->con);
  $r=@mysql_fetch_array($q);

  return $r;
 }

 function Update($aid,$name,$email)
 {
  $q=@mysql_query("update admins set Name='$name',Email='$email' where AID=$aid",$this->con);

  return $q;
 }

 function Modify_Password($aid,$password)
 {
  $q=@mysql_query("update admins set UPass=md5('$password') where AID=$aid",$this->con);
  return $q;

 }


}

?>
```

includes/db.php

```
<?php

class db
{
  var $server;
  var $user;
  var $pass;
  var $db;
  var $connection;

  function db($server="localhost",$user="root",$pass="",$db="pds")
  {
    $this->server=$server;
    $this->user=$user;
    $this->pass=$pass;
    $this->db=$db;
  }

  function connect()
  {
    $this->connection=mysql_connect($this->server,$this->user,$this->pass);
    mysql_select_db($this->db,$this->connection);

    return $this->connection;
  }
}


?>
```

includes/device.php

```
<?php

class Device
{
 var $SID;
 var $OwnerID;
 var $State;
 var $Con;

 function Device($SID="",$OwnerID="")
 {
  $this->SID=$SID;
  $this->OwnerID=$OwnerID;


  $db=new db();
  $this->Con=$db->connect();

 }

 function Add()
 {
  $q=mysql_query("select * from devices where SID='$this->SID'",$this->Con);
  $t=mysql_fetch_array($q);

  if($t)
  {
   return "EXIST";

  }
  else
  {
   $q2=mysql_query("insert into devices values('$this->SID',$this->OwnerID,'OK')",$this->Con);
   if($q)
    return "OK";
   else
    return "ERROR";
  }
}

function Suspend()
{
 $q=mysql_query("update devices set State='Suspended' where SID='$this->SID'",$this->Con);
 return $q;
}
```

```
 function unSuspend()
 {
  $q=mysql_query("update devices set State='OK' where SID='$this->SID'",$this->Con);
  return $q;
 }

 function Check()
 {
  $q=mysql_query("select * from devices where SID='$this->SID'",$this->Con);

  if(mysql_num_rows($q)!=0)
  {
   $q=mysql_query("select State from devices where SID='$this->SID'",$this->Con);
   $r=mysql_fetch_array($q);

   if($r['State']=="OK")
    return "O";//ok
   else
    return "S";// suspended
  }
  else
  {
   return "N";//not added
  }
 }

 function Show($SID,$state="")
 {
  $s="select * from devices,owners where devices.OwnerID=owners.OID and ";
  if($SID)
   $s=$s." devices.SID='$SID'";
  else
   $s=$s." devices.State like '$state'";

  $q=mysql_query($s,$this->Con);
  return $q;
 }
}

?>
```

includes/directory.php

```
<?php

class Ph_Directory
{
  var $fname;
  var $sname;
  var $tname;
  var $lname;
  var $address;
  var $phoneno;
  var $Con;

  function Ph_Directory($fname,$sname,$tname,$lname,$phoneno="",$address="")
  {
    $this->fname=strtoupper($fname);
    $this->sname=strtoupper($sname);
    $this->tname=strtoupper($tname);
    $this->lname=strtoupper($lname);
    $this->address=$address;
    $this->phoneno=$phoneno;

    $db=new db();
    $this->Con=$db->connect();
  }

  function Add()
  {
    $q=mysql_query("select * from directory where PhoneNo='$this->phoneno'",$this->Con);
    if(mysql_num_rows($q)>0)
      return "EXIST";

    $q=mysql_query("insert into directory values('$this->fname','$this->sname','$this->tname','$this-
>lname','$this->address','$this->phoneno')",$this->Con);
    if($q)
      return "OK";
    else
      return "ERROR";
  }

  function Perform_Query()// the result is for the pdd
  {
    $q=mysql_query("select * from directory where FName='$this->fname' and SName LIKE '$this-
>sname' and TName LIKE '$this->tname' and LName='$this->lname'",$this->Con);

    return $q;
```

```
  }

  function Show() // the result is to be viewed in CP
  {
    $q=mysql_query("select * from directory where FName like '$this->fname' and SName like '$this-
>sname' and TName like '$this->tname' and LName like '$this->lname' and PhoneNo like '$this-
>phoneno'",$this->Con);
    return $q;
  }
}


?>
```

includes/owner.php

```
<?php

class Owner
{
  var $name;
  var $phone;
  var $email;
  var $address;
  var $Con;

  function Owner($name="",$phone="",$email="",$address="")
  {
    $this->name=$name;
    $this->phone=$phone;
    $this->email=$email;
    $this->address=$address;

    $db=new db();
    $this->Con=$db->connect();
  }

  function Add()
  {
    $q=mysql_query("insert into owners(Name,PhoneNo,Email,Address) values('$this->name','$this->phone','$this->email','$this->address')",$this->Con);
    return $q;
  }

  function Update($OID)
  {
    $q=mysql_query("update owners set Name='$this->name',PhoneNo='$this->phone',Email='$this->email',Address='$this->address' where OID=$OID",$this->Con);

  }

  function Delete($OID)
  {
    $q=mysql_query("delete from owners where OID=$OID",$this->Con);
  }

  function Search()
  {
    $q=mysql_query("select * from owners where Name like '%$this->name%'",$this->Con);
    return $q;
  }
```

```
 function Show($OID="")
 {
  if($OID)
    $q=mysql_query("select * from owners where OID=$OID",$this->Con);
  else
    $q=mysql_query("select * from owners",$this->Con);
  return $q;
 }
}


?>
```

admin/index.php

```
<?php
session_start();
//session_destroy();
if(!@session_is_registered('admin_pds_772_x')||!@session_is_registered('aid_pds')||!@session_is_register
ed('name_pds'))
  echo "<meta http-equiv='refresh' content='0;url=login.php'>";
else
{
include("../includes/db.php");
include("../includes/owner.php");
include("../includes/device.php");
include("../includes/admin.php");
include("../includes/directory.php");
include("../includes/log.php");

?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<script language="javascript" src="functions.js"></script>
<title>Administration Control Panel</title>
</head>

<body>
<div class="container">
  <div class="header">
    <div class="header-cont">
      Phone Directory System
    </div>
  </div>
  <div class="sub-header">
  Administration Control Panel
  </div>
  <div class="sub2-header">
  <a href="index.php"><img src="images/home.png" width="50" height="50" alt="home" /></a>
  <a href="logout.php"><img src="images/logout.png" width="50" height="50" alt="logout" /></a>
  </div>

  <div class="left">
    <div class="items-title">
    PDDs Management
    </div>
```

```
<a href="?task=add_pdd"><div class="item">
Add New PDD
</div></a>
<a href="?task=find_pdds"><div class="item">
Find PDDs
</div></a>
<div class="items-title">
PDD Owners Management
</div>
<a href="?task=add_owner"><div class="item">
Add Owner
</div></a>
<a href="?task=find_owners"><div class="item">
Find Owners
</div></a>
<div class="items-title">
Phone Directory Management
</div>
<a href="?task=add_entry"><div class="item">
Add New Entry
</div></a>
<a href="?task=find_entry"><div class="item">
Find Entry
</div></a>
<div class="items-title">
Search Log
</div>
<a href="?task=show_log"><div class="item">
Show Search Log
</div></a>

<div class="items-title">
Admins Management
</div>
<a href="?task=modify_account"><div class="item">
Modify my account details
</div></a>
<a href="?task=add_admin"><div class="item">
 Add New Admin
</div></a>


</div>
<div class="right">
<?php
$task=@$_GET['task'];
```

```
 if($task=="")
 {
   echo "<h4>Welcome in adminstration control panel</h4>Here you can manage the Phone Directory
System.";
 }
 else if($task=="add_pdd")
 {
 ?>
  <div class="right-title">
  Add PDD
  </div><br/>
  <form name="add_pdd" action="?task=add_pdd_step2" method="post" onsubmit="return
validate_SID('add_pdd')">
  <table width="300" border="0" cellspacing="2" cellpadding="2">
 <tr>
  <td width="138" class="title">SID :</td>
  <td width="148">
   <input name="SID" type="text" id="textfield" maxlength="10" /></td>
 </tr>
 <tr>
  <td class="title">Owner :</td>
  <td><select name="oid">
  <?php
 $o=new Owner();
 $r=$o->Show();

 while($res=mysql_fetch_array($r))
 {
     echo "<option value=".$res['OID'].">".$res['Name']."</option>";
 }
 ?>

  </select></td>
 </tr>
<tr>
  <td colspan="2"><input type="submit" name="Add" id="button" value="Add" /></td>
 </tr>
</table>

  </form>
  <?php
 }// end else if add pdd
 else if($task=="add_pdd_step2")
 {
 ?>
```

```
?>
 <div class="right-title">
 Find PDDs
 </div><br/>
 <?php
$m=$_GET['method'];
$d=new Device();

  if($m=="SID")
    $q=$d->Show($_POST['SID']);
  else if($m=="all")
    $q=$d->Show("","%%");
  else
    $q=$d->Show("",$m);
?>
 <table class="center" width="100%" border="0" cellspacing="2" cellpadding="2">
 <tr >
 <td class="title">SID</td>
 <td class="title">Owner</td>
 <td class="title">Status</td>
 <td class="title">un/Suspend</td>
 <td class="title">Show Log</td>
</tr>
 <?php
  while($r=mysql_fetch_array($q))
  {
?>


<tr>
  <td><?php echo $r['SID']; ?></td>
  <td><a href="?task=find_owners_step2&OID=<?php echo $r['OID']; ?>" ><?php echo $r['Name'];
?></a></td>
  <td><?php if($r['State']=="OK")
        echo "<div class='green'>";
      else
        echo "<div class='red'>";
      echo $r['State']; ?></td></div></td>
  <td>
  <?php if($r['State']=="OK"){?>
  <a onclick="return confirm('Are you sure that you want to suspend this PDD')"
href="?task=suspend&SID=<?php echo $r['SID']; ?>"><img src="images/suspend.png" width="20"
height="20" alt="suspend" /></a>
  <?php }else { ?>
  <a onclick="return confirm('Are you sure that you want to unsuspend this PDD')"
href="?task=unsuspend&SID=<?php echo $r['SID']; ?>"><img src="images/unsuspend.png" width="20"
```

```
height="20" alt="suspend" /></a>
  <?php } ?>
  </td>
  <td><a  href="?task=show_log_step2&SID=<?php echo $r['SID']; ?>"><img src="images/log.gif"
width="20" height="20" alt="show log" /></a></td>
 </tr>


  <?php
   }// end while
 ?>
 </table>
 <?php
 }
 else if($task=="suspend")
 {
  $sid=$_GET['SID'];
  $d=new Device($sid);
  $r=$d->Suspend();

  if($r)
    echo "Suspended successfully.";
  else
    echo "An error happened.";
 }
 else if($task=="unsuspend")
 {
  $sid=$_GET['SID'];
  $d=new Device($sid);
  $r=$d->unSuspend();

  if($r)
    echo "unSuspended successfully.";
  else
    echo "An error happened.";
 }
 else if($task=="add_owner")
 {
 ?>
 <div class="right-title">
 Add PDD Owner
 </div><br/>
 <form name="add_owner" action="?task=add_owner_step2" method="post" onsubmit="return
validate_owner('add_owner')">
 <table width="300" border="0" cellspacing="2" cellpadding="2">
<tr>
```

```
  <td width="138" class="title">Name :</td>
  <td width="148">
    <input name="name" type="text" id="textfield" maxlength="100" /></td>
 </tr>
 <tr>
  <td class="title">Phone :</td>
  <td><input name="phone" type="text" id="textfield" maxlength="50" /></td>
 </tr>
 <tr>
  <td class="title">Email :</td>
  <td><input name="email" type="text" id="textfield" maxlength="100" /></td>
 </tr>
 <tr>
  <td class="title">Address</td>
  <td><input name="address" type="text" id="textfield" maxlength="100" /></td>
 </tr>
 <tr>
  <td colspan="2"><input type="submit" name="Add" id="button" value="Add" /></td>
 </tr>
</table>

  </form>

  <?php
  }
  else if($task=="add_owner_step2")
  {
  ?>
  <div class="right-title">
  Add PDD Owner
  </div><br/>

  <?php
    $name=$_POST['name'];
    $phone=$_POST['phone'];
    $email=$_POST['email'];
    $address=$_POST['address'];

    $o=new Owner($name,$phone,$email,$address);
    $r=$o->Add();
    if($r)
      echo "Added Successfully.";
    else
      echo "An error happened.";
  }
  else if($task=="find_owners")
```

```
  {
  ?>
   <div class="right-title">
   Find Owners
   </div><br/>
   Search for Owner by name :<br/>
   <form name="find_owners" action="?task=find_owners_step2" method="post">
   <input name="name" type="text" size="20" maxlength="100" /><input name="" type="submit"
value="Search" />
   </form>

   <div class="red">** If you leave the name field empty, you will get all owners as a result.</div>
   <?php
  } else if($task=="find_owners_step2")
  {
    $name=@$_POST['name'];
    if($name=="")
      $name="%%";

    $oid=@$_GET['OID'];

    $o=new Owner($name);
    if($oid)
      $q=$o->Show($oid);
    else
      $q=$o->Search();

    ?>
     <div class="right-title">
      Find Owners
      </div><br/>

     Search for Owner by name :<br/>
   <form name="find_owners" action="?task=find_owners_step2" method="post">
   <input name="name" type="text" size="20" maxlength="100" /><input name="" type="submit"
value="Search" />
   </form>

   <div class="red">** If you leave the name field empty, you will get all owners as a result.</div>
     <table class="center" width="100%" border="0" cellspacing="2" cellpadding="2">
   <tr >
   <td class="title">Name</td>
   <td class="title">Phone</td>
   <td class="title">Email</td>
   <td class="title">Address</td>
  </tr>
```

```
    <?php
   while($r=mysql_fetch_array($q))
   {
   ?>
    <tr>
    <td><?php echo $r['Name']; ?></td>
    <td><?php echo $r['PhoneNo']; ?></td>
    <td><?php echo $r['Email']; ?></td>
    <td><?php echo $r['Address']; ?></td>
    </tr>
    <?php
   }//end while
   ?>
    </table>
    <?php
 }
 else if($task=="add_entry")
 {
 ?>
  <div class="right-title">
  Add New Entry
  </div><br/>

  <form name="add_entry" action="?task=add_entry_step2" method="post" onsubmit="return
validate_entry('add_entry')">
  <table width="300" border="0" cellspacing="2" cellpadding="2">
<tr>
  <td width="138" class="title">First Name :</td>
  <td width="148">
    <input name="fname" type="text" id="textfield" maxlength="20" /></td>
</tr>
<tr>
  <td class="title">Second Name :</td>
  <td><input name="sname" type="text" id="textfield" maxlength="20" /></td>
</tr>
<tr>
  <td class="title">Third Name :</td>
  <td><input name="tname" type="text" id="textfield" maxlength="20" /></td>
</tr>
<tr>
  <td class="title">Last Name :</td>
  <td><input name="lname" type="text" id="textfield" maxlength="20" /></td>
</tr>
<tr>
  <td class="title">Address :</td>
  <td><input name="address" type="text" id="textfield" maxlength="100" /></td>
```

```
 </tr>
 <tr>
  <td class="title">Phone Number :</td>
  <td><input name="phoneno" type="text" id="textfield" maxlength="50" /></td>
 </tr>
 <tr>
  <td colspan="2"><input type="submit" name="Add" id="button" value="Add" /></td>
 </tr>
</table>

  </form>

  <?php
 }
 else if($task=="add_entry_step2")
 {
   $fname=$_POST['fname'];
   $sname=$_POST['sname'];
   $tname=$_POST['tname'];
   $lname=$_POST['lname'];
   $address=$_POST['address'];
   $phoneno=$_POST['phoneno'];

   $d=new Ph_Directory($fname,$sname,$tname,$lname,$phoneno,$address);
   $r=$d->Add();

   if($r=="EXIST")
     echo "This entry is already added.";
   else if($r=="OK")
     echo "Added succesfuly.";
   else if($r=="ERROR")
     echo "An error happened.";

 }
 else if($task=="find_entry")
 {
 ?>
  <div class="right-title">
  Find Entry
  </div><br/>

  <form action="?task=find_entry_step2" method="post">
  <table width="300" border="0" cellspacing="2" cellpadding="2">
<tr>
  <td width="138" class="title">First Name :</td>
  <td width="148">
```

```
    <input name="fname" type="text" id="textfield" maxlength="20" /></td>
  </tr>
  <tr>
   <td class="title">Second Name :</td>
   <td><input name="sname" type="text" id="textfield" maxlength="20" /></td>
  </tr>
  <tr>
   <td class="title">Third Name :</td>
   <td><input name="tname" type="text" id="textfield" maxlength="20" /></td>
  </tr>
  <tr>
   <td class="title">Last Name :</td>
   <td><input name="lname" type="text" id="textfield" maxlength="20" /></td>
  </tr>
  <tr>
   <td class="title">Phone Number :</td>
   <td><input name="phoneno" type="text" id="textfield" maxlength="50" /></td>
  </tr>
  <tr>
   <td colspan="2"><input type="submit" name="Add" id="button" value="Search" /></td>
  </tr>
 </table>

  </form>

  <div class="red">** You can leave any of these fields empty.</div>
  <div class="red">** If you leave all fields empty, you will get the whole directory as a result.</div>
  <?php
 }
 else if($task=="find_entry_step2")
 {
 ?>
  <div class="right-title">
  Find Entry
  </div><br/>
  <?php
   $fname=$_POST['fname'];
   $sname=$_POST['sname'];
   $tname=$_POST['tname'];
   $lname=$_POST['lname'];
   $phoneno=$_POST['phoneno'];

   if($fname=="")
     $fname="%%";
   if($sname=="")
     $sname="%%";
```

```
    if($tname=="")
      $tname="%%";
    if($lname=="")
      $lname="%%";
    if($phoneno=="")
      $phoneno="%%";
    $d=new Ph_Directory($fname,$sname,$tname,$lname,$phoneno);
    $r=$d->Show();



 ?>
  <table width="100%" border="0" cellspacing="2" cellpadding="2">
  <tr >
  <td class="title">First</td>
  <td class="title">Second</td>
  <td class="title">Third</td>
  <td class="title">Last</td>
  <td class="title">Address</td>
  <td class="title">Phone No.</td>
</tr>
  <?php
    while($e=mysql_fetch_array($r))
    {
 ?>


<tr>
  <td><?php echo $e['FName']; ?></td>
  <td><?php echo $e['SName']; ?></td>
  <td><?php echo $e['TName']; ?></td>
  <td><?php echo $e['LName']; ?></td>
  <td><?php echo $e['Address']; ?></td>
  <td><?php echo $e['PhoneNo']; ?></td>
</tr>


 <?php
   }// end while
?>
 </table>
 <?php
}
else if($task=="show_log")
{
?>
```

```
   <div class="right-title">
   Show Search Log
   </div><br/>
   Show search log for a specific PDD :<br/>
   <form action="?task=show_log_step2" method="post">
   <input name="SID" type="text" size="14" maxlength="10" /><input name="" type="submit"
value="Search" />
   </form>
  <div class="red">** If you leave the SID field empty, you will get the whole search log as a
result.</div>

   <?php
  }
  else if($task=="show_log_step2")
  {
  ?>
    <div class="right-title">
   Show Search Log
   </div><br/>
   Show search log for a specific PDD :<br/>
   <form action="?task=show_log_step2" method="post">
   <input name="SID" type="text" size="14" maxlength="10" /><input name="" type="submit"
value="Show" />
   </form>
  <div class="red">** If you leave the SID field empty, you will get the whole search log as a
result.</div>
   <?php

  $sid=@$_GET['SID'];
  if(!$sid)
    $sid=$_POST['SID'];
  $l=new Log();
  $q=$l->Search($sid);
  ?>
  <table class="center" width="100%" border="0" cellspacing="2" cellpadding="2">
  <tr >
  <td class="title">SID</td>
  <td class="title">IP</td>
  <td class="title">Date</td>
  <td class="title">Looked for</td>
</tr>
    <?php
   while($r=mysql_fetch_array($q))
   {
   ?>
```

```
      <tr>
      <td><?php echo $r['SID']; ?></td>
      <td><?php echo $r['IP']; ?></td>
      <td><?php echo $r['Date']; ?></td>
      <td><?php echo $r['LookedFor']; ?></td>
      </tr>
      <?php
     }//end while
     ?>
      </table>
   <?php



   }
   else if($task=="modify_account")
   {
     $a=new Admin();
     $r=$a->Show_Admin($_SESSION['aid_pds']);
   ?>
    <div class="right-title">
    Modify My Account
    </div><br/>

   <br/>
    <form name="account_mod" action="?task=modify_account_step2" method="post" onsubmit="return
 validate_account('account_mod')">
    <table width="300" border="0" cellspacing="2" cellpadding="2">
  <tr>
    <td width="138" class="title">User Name :</td>
    <td width="148">
     <input name="uname" type="text" id="textfield" value="<?php echo $r['UName']; ?>"
maxlength="50" readonly="readonly"/></td>
  </tr>
  <tr>
    <td class="title">Password :</td>
    <td><input name="pass" type="password" id="textfield" maxlength="20" /></td>
  </tr>
  <tr>
    <td class="title">Name :</td>
    <td><input name="name" type="text" id="textfield" value="<?php echo $r['Name']; ?>"
maxlength="100" /></td>
  </tr>
  <tr>
    <td class="title">Email :</td>
    <td><input name="email" type="text" id="textfield"  value="<?php echo $r['Email']; ?>"
```

```
maxlength="100"/></td>
 </tr>
 <tr>
 <tr>
  <td colspan="2"><input type="submit" name="Add" id="button" value="Modify" /></td>
 </tr>
</table>

  </form>
  <div class="red">** Leave the password field empty if you don't want to modify.</div>
  <?php
 }
 else if($task=="modify_account_step2")
 {
   $pass=$_POST['pass'];
   $name=$_POST['name'];
   $email=$_POST['email'];

   $a=new Admin();
   $r1=$a->Update($_SESSION['aid_pds'],$name,$email);
   if($pass)
     $r2=$a->Modify_Password($_SESSION['aid_pds'],$pass);

   if($pass)
   {
     if($r1&&$r2)
       echo "Modified Successfully.";
     else
       echo "An Error Happened";
   }
   else
   {
     if($r1)
       echo "Modified Successfully.";
     else
       echo "An Error Happened";
   }
 }
 else if($task=="add_admin")
 {
 ?>
 <div class="right-title">
 Add New Admin
 </div><br/>

 <br/>
```

```
  <form name="add_admin" action="?task=add_admin_step2" method="post" onsubmit="return
validate_add_admin('add_admin')">
  <table width="300" border="0" cellspacing="2" cellpadding="2">
 <tr>
  <td width="138" class="title">User Name :</td>
  <td width="148">
   <input name="uname" type="text" id="textfield" maxlength="50" /></td>
 </tr>
 <tr>
  <td class="title">Password :</td>
  <td><input name="pass" type="password" id="textfield" maxlength="20" /></td>
 </tr>
 <tr>
  <td class="title">Confirm Password :</td>
  <td><input name="cpass" type="password" id="textfield" maxlength="20" /></td>
 </tr>

 <tr>
  <td class="title">Name :</td>
  <td><input name="name" type="text" id="textfield" maxlength="100" /></td>
 </tr>
 <tr>
  <td class="title">Email :</td>
  <td><input name="email" type="text" id="textfield" maxlength="100" /></td>
 </tr>
 <tr>
 <tr>
  <td colspan="2"><input type="submit" name="Add" id="button" value="Add" /></td>
 </tr>
</table>

  </form>
  <?php
 }
 else if($task=="add_admin_step2")
 {
   $uname=$_POST['uname'];
   $pass=$_POST['pass'];
   $name=$_POST['name'];
   $email=$_POST['email'];

   $a=new Admin($uname,$pass,$name,$email);
   $r=$a->Create_Admin();

   if($r=="existuser")
     echo "This Username is already registered.";
```

```
    else if($r=="existemail")
      echo "This email is already registered.";
    else if($r=="done")
      echo "Admin added successfully.";
    else if($r=="error")
      echo "An error happened.";
  }

  ?>
  </div>

  <div class="footer">
  All Rights Are Reserverd.
  </div>
</div>
</body>
</html>
<?php } ?>
```

admin/login.php

```
<?php
session_start();

if(@session_is_registered('admin_pds_772_x')&&@session_is_registered('aid_pds')&&@session_is_regis
tered('name_pds'))
  echo "<meta http-equiv='refresh' content='0;url=index.php'>";
else
{
include("../includes/db.php");
include("../includes/admin.php");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<title>Administration Login</title>
</head>

<body>
<div class="login-container">
<?php
$task=@$_GET['task'];

if($task=="")
{
?>
    <div class="login-image">
  <img src="images/Lock-icon.png" width="150" height="150" alt="pds" />
  </div>
    <div class="login-box">
    <form action="?task=login" method="post" name="form1">
    <table  width="100%" border="0" cellspacing="0" cellpadding="0">
    <tr>
    <td class="noborder" colspan="2"><h4>Administration Login</h4></td>
    </tr>
     <tr>
      <td class="noborder">User Name :</td>
      <td class="noborder"><input name="uname" type="text" /></td>
     </tr>
     <tr>
      <td class="noborder">Password :</td>
      <td class="noborder"><input name="upass" type="password" /></td>
     </tr>
```

```
        <tr>
          <td class="noborder"> </td>
          <td class="noborder"><input name="Login" type="submit" value="Login" /></td>
        </tr>
      </table>


      </form>
<?php
}
else if($task=='login')
{
$admin_pds_772_x=$_POST['uname'];
$upass=$_POST['upass'];

$u=new Admin($admin_pds_772_x,$upass);
$r=$u->Is_Admin();

  if($r)
  {
    $aid_pds=$r['AID'];
    $name_pds=$r['Name'];

    @session_register('admin_pds_772_x');
    @session_register('name_pds');
    @session_register('aid_pds');

    echo "<br/><br/><center><h4>Logged in successfully<br/>You will be redirected to your control
panel</h4></center>";
    echo "<meta http-equiv='refresh' content='3;url=index.php'><br/>";
  }
  else
  {
    echo "<br/><br/><center><h4>Wrong username or password</h4></center><br/><br/>";
  }

}
?>
    </div>
  </div>
</body>
</html>
<?php } ?>
```

admin/logout.php

```
<?php
session_start();
session_destroy();

echo "<meta http-equiv='refresh' content='0;url=login.php'>";
?>
```

admin/functions.js

```
// JavaScript Document

function validate_SID(form_name)
{
  if(form_name=="add_pdd")
    sid=document.add_pdd.SID.value;
  else if(form_name=="find_pdd")
    sid=document.find_pdd.SID.value;

  if(sid.length!=10)
  {
    alert("SID should be 10 characters long");
    return false;
  }
  else
    return true;
}

//=================================================
function validate_owner(form_name)
{
  if(form_name=="add_owner")
    f=document.add_owner;

  name=f.name.value;
  phone=f.phone.value;
  email=f.email.value;
  address=f.address.value;

  if(name==""||phone==""||email==""||address=="")
    {
      alert("All fields should be filled");
      return false;
    }

  if(email.indexOf('@')==-1)
    {
      alert("Invalid email address");
      return false;
    }
  return true;
}
//=================================================
function validate_entry(form_name)
{
```

```
  if(form_name=="add_entry")
    f=document.add_entry;

  fname=f.fname.value;
  sname=f.sname.value;
  tname=f.tname.value;
  lname=f.lname.value;
  address=f.address.value;
  phoneno=f.phoneno.value;

  if(fname==""||sname==""||tname==""||lname==""||address==""||phoneno=="")
  {
    alert("All fields should be filled");
    return false;
  }

  return true;
}
//=========================================
function validate_account(form_name)
{
  if(form_name=="account_mod")
    f=document.account_mod;
  else if(form_name=="add_admin")
    f=document.add_admin;

  name=f.name.value;
  email=f.email.value;
  if(name=="")
  {
    alert("The name field should be filled");
    return false;
  }

 if(email=="")
 {
   alert("The email field should be filled");
   return false;
 }

 if(email.indexOf('@')==-1)
   {
     alert("Invalid email address");
     return false;
   }
```

```
  return true;
}
//=========================================
function validate_add_admin(form_name)
{
  uname=document.add_admin.uname.value;
  pass=document.add_admin.pass.value;
  cpass=document.add_admin.cpass.value;

  if(uname=="")
  {
    alert("User name field should be filled");
    return false;
  }

  if(pass=="")
  {
    alert("Password field should be filled");
    return false;
  }

  if(pass!=cpass)
  {
    alert("Password confirmation is wrong");
    return false;
  }
  if(validate_account(form_name))
    return true;
  else return false;
}
```

admin/css/style.css

```
@charset "utf-8";
/* CSS Document */

body
{
  text-align:center;
  margin:0px;

}

.container
{
  margin-left:auto;
  margin-right:auto;
  width:1000px;
  text-align:left;
  border-style:double;
  border-width:2px;
  overflow:hidden;
}

.header
{
  height:80px;
  background-image:url(../images/fbg.jpg);
  background-repeat:repeat-x;
  border-bottom-style:double;
  border-width:2px;

}
.header-cont
{
  color:#FFF;
  font-size:28px;
  padding:19px;
  font-weight:bold;
}

.sub-header
{
  background-image:url(../images/gradient_tcat.jpg);
  background-repeat:repeat-x;
  font-size:24px;
  color:#FFF;
  font-weight:bold;
```

```
}

.sub2-header
{
  height:52px;
  background-color:#F0F0F0;
  border-bottom-style:double;
  border-width:2px;
}

.left
{
  float:left;
  width:195px;
  border-right-style:double;
  /*border-left-style:double;*/
  border-width:2px;
  margin-top:1px;
  /*margin-left:5px;*/
}

.right
{
  float:left;
  width:776px;
  padding:10px;
  padding-top:1px;

}

.footer
{
  clear:both;
  background-image:url(../images/gradient_tcat.jpg);
  background-repeat:repeat-x;
  border-top-style:double;
  border-width:2px;
  border-color:#000;
  color:#FFF;
  text-align:center;

}

.items-title
{
  background-image:url(../images/gradient_tcat.jpg);
```

```
  background-repeat:repeat-x;
  color:#FFF;
  border-top-style:double;
  border-width:2px;
  border-color:#000;

}

.item
{
  background-color:#F0F0F0;
  border-bottom-style:dotted;
  border-width:1px;
}

.item:hover
{
  background-color:#787878;
}

.right-title
{
  font-size:18px;
  font-weight:bold;
}

table
{

}

td
{
  border-style:solid;
  border-width:1px;
  border-color:#999;
}

td.noborder
{
  border-style:none;
}
td.title
{
  background-color:#EEE;
}
```

```
a
{
  color:#006;
  text-decoration:none;
}

a:visited
{
  color:#006;
  text-decoration:none;
}

a:hover
{
  color:#000;
  text-decoration:none;
}

.red
{
  color:#F00;
}

.green
{
  color:#3C6;
}

img
{
  border:none;
}

.center
{
  text-align:center;
}
/*===*/
.login-container
{
  text-align:left;
  width:400px;
  height:160px;
  margin-left:auto;
  margin-right:auto;
```

```
  margin-top:100px;
  border-style:double;
  border-color:#666;
}

.login-image
{
  float:left;
  width:150px;
  margin-top:5px;
}

.login-box
{
  width:250px;
  float:left;
  margin-top:10px;
}
```