AN ABSTRACT OF THE THESIS OF

Hua Yu for the degree of Master of Science in Electrical and Computer Engineering presented on March 1, 1989.

Title: EDLAN: An Educational Local Area Network

Abstract approved: Redacted for Privacy

James H. Herzog

EDLAN is a simple, inexpensive local area network suitable for educational applications. In EDLAN, a number of personal computers are connected through an RS-485 global bus which carries the data transmission among the computers. CSMA/CD protocol is used to control the access of the global bus. The communications are controlled by 8051 single board computer based Network Interface Units attached to each personal computer.

In this network, one computer is used by an instructor and the rest used by students. Class and exam facilities are developed so that the instructor can use the network to teach classes, to distribute class materials, exam or exercise problems. The students can use the network to take classes or exams without physically going to the same classroom.

Housekeeping and grading functions are provided to help the instructor to manage the host computer and to manage classes and examinations. Friendly menu driven operation, on-line help facility, and window functions make the system easy to use.
EDLAN: An Educational Local Area Network

by

Hua Yu

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirement for the
degree of

Master of Science

Completed March 1, 1989

Commencement June, 1989
APPROVED:

Redacted for Privacy
Associate Professor of Electrical and Computer Engineering in charge of major

Redacted for Privacy
Head of Department of Electrical and Computer Engineering

Redacted for Privacy
Dean of Graduate School

Date thesis is presented March 1, 1989

Typed by Hua Yu for Hua Yu
ACKNOWLEDGEMENTS

I would like to express my heartfelt thanks to a number of people for their help in the preparation of this thesis. Foremost among them is my major advisor, Dr. James H. Herzog, whose guidance during all my graduate studies has been invaluable. I would also like to thank my thesis committee, Professor Leland Jensen, Professor Donald Amort, Professor David Ullman and Professor W. J. Kolodziej for their fruitful suggestions and thorough thesis reading.

Most of all, I wish to thank all my family members, my parents, my husband and my son, whose constant support and encouragement make this work possible.
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Computer Aided Instruction</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Objective of the Thesis</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Outline of the Following Chapters</td>
<td>4</td>
</tr>
<tr>
<td>2. LOCAL AREA NETWORKS</td>
<td>6</td>
</tr>
<tr>
<td>2.1 Application of Computers</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Computer Networks</td>
<td>6</td>
</tr>
<tr>
<td>2.3 Network Topologies</td>
<td>7</td>
</tr>
<tr>
<td>2.4 Medium Access Protocol</td>
<td>9</td>
</tr>
<tr>
<td>2.4.1 Token Bus Protocol</td>
<td>10</td>
</tr>
<tr>
<td>2.4.2 CSMA/CD Protocol</td>
<td>10</td>
</tr>
<tr>
<td>2.5 ISO Network Model</td>
<td>11</td>
</tr>
<tr>
<td>2.6 Serial Data Transmission Protocols</td>
<td>14</td>
</tr>
<tr>
<td>3. NETWORK INTERFACE UNIT AND COMMUNICATION PROTOCOL</td>
<td>18</td>
</tr>
<tr>
<td>3.1 Overview</td>
<td>18</td>
</tr>
<tr>
<td>3.2 Network Interface Unit</td>
<td>18</td>
</tr>
<tr>
<td>3.2.1 Central Processing Unit</td>
<td>18</td>
</tr>
<tr>
<td>3.2.2 Interface with Host Computers</td>
<td>19</td>
</tr>
<tr>
<td>3.2.3 Node Address Setting and Print Port</td>
<td>19</td>
</tr>
<tr>
<td>3.2.4 Global Bus Interface Circuits</td>
<td>20</td>
</tr>
<tr>
<td>3.3 Communication Data Format</td>
<td>21</td>
</tr>
<tr>
<td>3.3.1 Control Command Packet</td>
<td>22</td>
</tr>
<tr>
<td>3.3.2 Message Packet</td>
<td>23</td>
</tr>
</tbody>
</table>
# TABLE OF CONTENTS (Continuation)

3.4 Summaries

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. IMPLEMENTATION OF THE EDLAN SOFTWARE</td>
<td>29</td>
</tr>
<tr>
<td>4.1 Overview</td>
<td>29</td>
</tr>
<tr>
<td>4.2 Communication Functions</td>
<td>30</td>
</tr>
<tr>
<td>4.2.1 Serial Data Buffer</td>
<td>30</td>
</tr>
<tr>
<td>4.2.2 Serial Data Transfer Procedures</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Housekeeping Functions</td>
<td>35</td>
</tr>
<tr>
<td>4.3.1 File and Directory Operation Functions</td>
<td>35</td>
</tr>
<tr>
<td>4.3.2 Screen Editor</td>
<td>36</td>
</tr>
<tr>
<td>4.4 Class and Exam Functions</td>
<td>39</td>
</tr>
<tr>
<td>4.4.1 Open Class and Exam Functions</td>
<td>39</td>
</tr>
<tr>
<td>4.4.2 InClass and InExam Functions</td>
<td>41</td>
</tr>
<tr>
<td>4.4.3 Grading Functions</td>
<td>46</td>
</tr>
<tr>
<td>4.5 Miscellaneous Functions</td>
<td>48</td>
</tr>
<tr>
<td>4.5.1 On-line Help Function</td>
<td>48</td>
</tr>
<tr>
<td>4.5.2 System Reset Functions</td>
<td>48</td>
</tr>
<tr>
<td>4.5.3 Menu Selection Functions</td>
<td>48</td>
</tr>
<tr>
<td>4.5.4 Window Management Functions</td>
<td>50</td>
</tr>
<tr>
<td>4.6 Summaries</td>
<td>50</td>
</tr>
<tr>
<td>5. SUMMARIES AND SUGGESTIONS FOR FUTURE WORK</td>
<td>59</td>
</tr>
<tr>
<td>5.1 Results and Summaries</td>
<td>59</td>
</tr>
<tr>
<td>5.2 Suggestions for Future Work</td>
<td>60</td>
</tr>
</tbody>
</table>

BIBLIOGRAPHY
## APPENDIX A  USER'S MANUAL FOR INSTRUCTOR

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1 Overview</td>
<td>66</td>
</tr>
<tr>
<td>A.2 Start of the Program</td>
<td>66</td>
</tr>
<tr>
<td>A.3 File Operations</td>
<td>67</td>
</tr>
<tr>
<td>A.3.1 Show Directory</td>
<td>67</td>
</tr>
<tr>
<td>A.3.2 Make Directory</td>
<td>68</td>
</tr>
<tr>
<td>A.3.3 Change Directory</td>
<td>68</td>
</tr>
<tr>
<td>A.3.4 Erase Directory</td>
<td>68</td>
</tr>
<tr>
<td>A.3.5 Delete File</td>
<td>68</td>
</tr>
<tr>
<td>A.3.6 Rename File</td>
<td>68</td>
</tr>
<tr>
<td>A.3.7 List File</td>
<td>68</td>
</tr>
<tr>
<td>A.3.8 Print File</td>
<td>69</td>
</tr>
<tr>
<td>A.4 Reset Operations</td>
<td>69</td>
</tr>
<tr>
<td>A.4.1 Set Date</td>
<td>69</td>
</tr>
<tr>
<td>A.4.2 Set Time</td>
<td>69</td>
</tr>
<tr>
<td>A.5 Class Operations</td>
<td>69</td>
</tr>
<tr>
<td>A.5.1 Open Class</td>
<td>70</td>
</tr>
<tr>
<td>A.5.2 InClass</td>
<td>71</td>
</tr>
<tr>
<td>A.5.3 Class Status</td>
<td>74</td>
</tr>
<tr>
<td>A.6 Exam Operations</td>
<td>74</td>
</tr>
<tr>
<td>A.6.1 Open Exam</td>
<td>74</td>
</tr>
<tr>
<td>A.6.2 InExam</td>
<td>74</td>
</tr>
<tr>
<td>TABLE OF CONTENTS (Continuation)</td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td></td>
</tr>
<tr>
<td>A.6.3 Grading .......................... 76</td>
<td></td>
</tr>
<tr>
<td>A.7 Edit Operation ...................... 77</td>
<td></td>
</tr>
<tr>
<td>A.8 Quit .................................. 78</td>
<td></td>
</tr>
<tr>
<td>APPENDIX B USER'S MANUAL FOR STUDENT</td>
<td></td>
</tr>
<tr>
<td>PROGRAM .................................. 85</td>
<td></td>
</tr>
<tr>
<td>B.1 Overview ............................ 85</td>
<td></td>
</tr>
<tr>
<td>B.2 Start of the Program ................. 85</td>
<td></td>
</tr>
<tr>
<td>B.3 Login .................................. 85</td>
<td></td>
</tr>
<tr>
<td>B.4 Taking a Class ....................... 86</td>
<td></td>
</tr>
<tr>
<td>B.5 Taking an Exam ...................... 87</td>
<td></td>
</tr>
<tr>
<td>B.6 Quit .................................. 89</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 EDLAN Structure</td>
<td>5</td>
</tr>
<tr>
<td>2.1 Computer Network Topologies</td>
<td>16</td>
</tr>
<tr>
<td>2.2 ISO OSI Layered Structure</td>
<td>17</td>
</tr>
<tr>
<td>3.1 Structure of Network Interface Unit</td>
<td>26</td>
</tr>
<tr>
<td>3.2 RS-485 Interface Circuit Diagram</td>
<td>27</td>
</tr>
<tr>
<td>3.3 Control Command Packet Format</td>
<td>28</td>
</tr>
<tr>
<td>3.4 Data Packet Format</td>
<td>28</td>
</tr>
<tr>
<td>4.1 Flowchart of the Instructor Main Program</td>
<td>51</td>
</tr>
<tr>
<td>4.2 Flowchart of Student Main Program</td>
<td>52</td>
</tr>
<tr>
<td>4.3 EDLAN Software Structure</td>
<td>53</td>
</tr>
<tr>
<td>4.4 Com1_buffer Structure</td>
<td>54</td>
</tr>
<tr>
<td>4.5 Format of the Messages Transmitted from NIU to the Host Computer</td>
<td>54</td>
</tr>
<tr>
<td>4.6 Flowchart for Check_buffer Routine in Instructor Program</td>
<td>55</td>
</tr>
<tr>
<td>4.7 Flowchart of the Check_buffer Routine in Student Program</td>
<td>56</td>
</tr>
<tr>
<td>4.8 Flowchart of the Inclass Procedure in Instructor Program</td>
<td>57</td>
</tr>
<tr>
<td>4.9 Flowchart of the Exam Operation in Student Program</td>
<td>58</td>
</tr>
<tr>
<td>Appendix A</td>
<td>Description</td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>A.1</td>
<td>Main Operation Menu for Instructor Program</td>
</tr>
<tr>
<td>A.2</td>
<td>File Operation Menu for Instructor Program</td>
</tr>
<tr>
<td>A.3</td>
<td>Class Operation Menu for Instructor Program</td>
</tr>
<tr>
<td>A.4</td>
<td>Class Status Window Format</td>
</tr>
<tr>
<td>A.5</td>
<td>InClass Operation Window Format for Instructor Program</td>
</tr>
<tr>
<td>A.6</td>
<td>Test(Exam) Operation Menu for Instructor program</td>
</tr>
<tr>
<td>A.7</td>
<td>Exam Type Selection Window Format</td>
</tr>
<tr>
<td>A.8</td>
<td>InExam Operation Window Format for Instructor Program</td>
</tr>
<tr>
<td>A.9</td>
<td>Example of Problem Files</td>
</tr>
<tr>
<td>B.1</td>
<td>Window Format for Student Login Session</td>
</tr>
<tr>
<td>B.2</td>
<td>Window Format for Student Class Session</td>
</tr>
<tr>
<td>B.3</td>
<td>Window Format for Student Exercise Session</td>
</tr>
<tr>
<td>B.4</td>
<td>Window Format for Student Question Session</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

1.1 Computer Aided Instruction

With the advent of low cost microcomputers, Computer Aided Instruction (CAI) has recently developed very quickly. According to surveys in project CASE (Computer Activity in Secondary Education) conducted by the American Institute for Research (AIR), 27 percent of secondary schools in the United States used computers for instructional applications in 1975. This number increased to 85 percent by 1982 [TOLMA84].

Today, computers have been successfully applied to educational systems [GODFR82, IRBY82, BECKE87]. However, in most schools, the CAI application is generally in extracurricular study activities or computer teaching laboratories in which teachers and students work on isolated computers [BECKE87]. This traditional CAI model has the following limitations:

1) It can not be used in a classroom environment to help instructors to teach regular classes.

2) Instructors and students must be physically in the same classroom to teach and take a class.
3) Instructors and students can not exchange class materials directly using their computers except by leaving messages on the same computer.

As the price reduction in computers, communication media and interfacing devices continues, it is both technically and economically feasible, at the present time, to link a number of small computers to construct a computer network for the educational application. This educational network overcomes the limitations of the conventional CAI model mentioned above. Instructors and students can exchange teaching and learning materials through the network. They can easily share their information and ideas with others through the network so that they can work and learn more efficiently.

At Oregon State University, a low cost, versatile local area network has been designed under the direct of Professor James H. Herzog. It evolved from Task Master [HERZO86] to COLAN1 [ZHENG86], COLAN2 [KAO87], COLAN3 [EUM87] and COLAN4 [THYE88]. Investigation shows that this network structure is appropriate for the CAI application. Its low cost and easy implementation make it very attractive for a class teaching environment. The design and implementation of the CAI application using the COLAN network structure will be the main topic of this thesis.

1.2 Objective of the Thesis

The objective of this project is to design the EDLAN, an Educational Local Area Network which has the functions and capabilities discussed in the previous section. The structure of this inexpensive local area network is shown in Figure 1.1.
The **EDLAN** can be used in a classroom environment. It links up to 63 personal computers through a global bus. Each personal computer is attached to a Network Interface Unit (NIU) which controls the data communication among the computers. Instructors and students use the computers in the network to exchange information, to teach classes and to take examinations.

More specifically, the instructor can use **EDLAN** to do the following:

- Supervise classes and examinations
- Distribute class materials and exam problems
- Develop class materials, exam problems and answer keys.
- Answer student questions
- Analyze student exam results
- Maintain class records

On the other side, the students can use the **EDLAN** to do the following:

- Log in a class or exam
- Receive class materials, exam or exercise problems
- Ask questions
- Solve exercise or exam problems
- Submit test results

**EDLAN** makes it possible, though not necessary, for students to take a class without physically going to the same classroom. They can take a class or an examination at a terminal located in different location. This is especially helpful to a handicapped student. He can even take a class at
home or in a hospital. On the instructor's side, the class management function, the grading and result analysis functions of EDLAN software relieve the instructors from many tedious tasks.

The EDLAN uses the low level communication functions in COLAN4 developed by Thye [THYE88] to supervise the data transmission among the Network Interface Units. The primary intention of this project is to develop the high level control functions to support the educational application and data transmission.

To ease the use of the EDLAN, the application programs are developed in a menu driven style. All the operations that can be performed at any time are displayed on the screen. They can be activated by pressing the corresponding keys. The window functions enable the users to display different messages in different windows and switch between windows. They make the application of EDLAN easier and more attractive.

1.3 Outline of the Following Chapters

Chapter 2 is an overview of computers and local area networks. It reviews the ISO OSI model, discusses the problem of network topology and medium access protocols.

Chapter 3 briefly discusses the structure of the Network Interface Unit and the functions it provides. The communication control mechanisms and data format are also discussed in this chapter.

Chapter 4 is dedicated to the implementation of the EDLAN control and application programs. It focuses on the technical design of the functions that interface with the Network Interface Units and with the host
computer operating systems. The functions that supervise classes, exams and other educational functions are also discussed in this chapter.

Chapter 5 summarizes the work in the EDLAN and makes some suggestions for future research.

User's manuals for the instructor's program and the student's program of the EDLAN are included as Appendix A and B, respectively.

---

**Figure 1.1 EDLAN Structure**

PC : Personal Computer
NIU : Network Interface Unit
CHAPTER 2

LOCAL AREA NETWORKS

2.1 Application of Computers

Since the emergence of ENIAC, the first electronic computer, in 1946, there have been continued developments in computer systems in terms of performance, applications and cost reduction.

Twenty years ago, computers could be afforded only by large organizations. They were mainly used for complex scientific computations and process control. With the rapid development in LSI, VLSI technology and in computer architecture, the performance of computers has been greatly improved and their price has quickly dropped. As a result of the improvement in the performance to cost ratio, computers now have much wider applications.

2.2 Computer Networks

One area that has developed extremely fast in the last two decades is computer network technology. A Computer Network is defined as a network in which a collection of computers, terminals and other peripheral devices are connected by communication channels. The communication channels can be pairs of twisted wires, coaxial cables, optical fibers, radio communication or even satellite links. The computers and terminals are capable of exchanging information with others through the communication media.
This combination of computer and communication technology provides each user with the ability to quickly and efficiently exchange information with other users. It enables a user in the network to access computing facilities, hardware and software at remote locations directly through his computer.

The emergence of computer networks opened many new research and application areas for computer technologies. It helps the computer industry to provide its users with more powerful, more reliable and lower cost computer facilities.

A computer network can be classified into two categories according to its geographical distribution, Wide Area Networks (WAN) or Local Area Networks (LAN). A wide area network is also called a long haul network. It is distributed in a wide area, nation wide or even world wide. On the other hand, a local area network is one that is distributed in a small geographical area. It is usually owned by a single organization [STALL87]. Because of its short distance, A LAN is usually characterized by[STALL87]:

- Low cost
- High transmission speed.
- Low transmission error rate.

2.3 Network Topologies

The term topology, in the context of computer networks, refers to the ways in which the computers are geographically connected by communication media. A topology is defined by the layout of the
communication media among the computers. It determines the data paths that may be used for data transmission between any pair of computers in the network.

One extreme of network topology is a completely connected network, or fully connected network. In this topology, a direct connection is provided between any pair of computers so that each computer can directly talk to another. However, the high cost and complexity limit its use in local area networks. For a network with N stations, we will need N*(N-1)/2 links to construct this completely connected network. Furthermore, the Network Interface Unit in each station must have (N-1) Input/Output ports to interface with the (N-1) links. Figure 2.1.a shows a completely connected network with six nodes.

A general mesh network is shown in Figure 2.1.b. In this network, a direct link can be put arbitrarily between any pair of computers. Some of the communication has to go through some intermediate nodes. These mesh structures are commonly used in wide area networks. They are very complex and expensive for local area networks. In a mesh network, each node may need to drive several Input/Output links. Furthermore, there may be more than one path to carry messages from one node to another. There must be some routing functions in each node to decide the path to use. The routing function is not a trivial problem[SCHWA80].

The most commonly used topologies in local area networks are bus and ring structures[BOULT85]. The structure of a ring network with 6 nodes is shown in Figure 2.1.c. Each node has two links connected to it. It is capable of receiving data on one link and transmitting on to the other, bit
by bit. At each node, the data is regenerated and delayed by one or more bits depending on the control protocol. The messages are transmitted through the ring in a single direction [IEEE85E,LIU83].

In a bus network as shown in Figure 2.1.d, all nodes are connected to a single global bus. All transmissions are through this bus. Routing is not needed because the bus is the only transmission medium. At any time, there is only one node allowed to transmit messages through the bus. The transmission is in a broadcasting style. The message can reach all the nodes connected to the bus. However, only the addressed destination node will receive the message. A bus network is better suited for the **EDLAN** than a ring network because of the following:

1) It is easy to delete or add nodes onto a bus.
2) Bus networks are more reliable than Ring networks. A single node failure may not have any effect on another node's operation as long as the instructor's node remains functional.
3) Bus networks are inherently more suitable for broadcast message transmission which is a major operation used in a classroom environment.

2.4 Medium Access Protocol

The bus in the network is a shared resource. All the data transmissions are carried out through it. Any node in the network may want to transmit messages at any time. When two or more nodes transmit messages simultaneously, the messages will collide with each other and all the messages involved in the collision will be damaged. There must be some mechanisms and rules to control the access of the bus so that messages can be reliably delivered to their destination. The set of such rules and
mechanisms is called the medium access protocol. There are currently two kinds of protocols commonly used in bus type networks, namely the Token Bus protocol and CSMA/CD protocol.

2.4.1 Token Bus Protocol

Token Bus is a collision free protocol in which collisions are avoided by a token mechanism [IEEE85D]. For this protocol, the node addresses are arranged in an ordered sequence with the last one followed by the first one. Thus, the nodes in the network form a logical ring.

A control frame called Token circulates in the network along the logical ring. The node receiving the token is granted control of the bus for message transmission. Upon finishing its transmission, the node passes the token to the next node along the logical ring. The next node now has the right to transmit. Nodes that do not have any message to transmit may pass the token directly to their successors. The control token is passed from node to node and every node in the network gets its opportunity to transmit.

2.4.2 CSMA/CD Protocol

The CSMA/CD (Carrier Sense Multiple Access with Collision Detection) protocol is currently the most commonly used medium access control technique for bus networks. It was first used by Xerox in its Ethernet local area network [METC76,SHOC82] and adopted by IEEE 802 committee as a standard for bus topology [IEEE85C].

In the CSMA/CD protocol, a node wishing to transmit first determines if there is any data being transmitted on the bus (Carrier Sensing). If no data transmission is detected, the node then immediately
starts message transmission. Otherwise, the node waits for a random amount of time and starts to sense the channel again. This process is repeated until the message is successfully transmitted.

However, starting transmission when the bus is idle does not always guarantee a successful delivery of the data to its destination. Consider the case that one node senses the channel idle and transmits a message. The data signal propagates to other nodes along the bus at a finite speed. Another node in the network may sense the channel before the first node's signal reaches it. The channel will be sensed idle by the second node. So the second node will start to transmit its message to the bus. A collision will occur in this case and all the data involved are damaged.

In a network using the **CSMA/CD** protocol, a node which transmits a message also simultaneously monitors the bus. A violation of the legal code word or a mismatch between the transmitted data and the received data will be recognized as a collision. The nodes involved in the collision terminate their transmission, wait for a random amount of time and then re-start the bus monitoring and message transmission process.

Collisions are allowed in **CSMA/CD** protocol. However, the collision detection and retransmission mechanisms ensure correct data deliveries.

2.5. ISO Network Model

To minimize design complexity, modern computer networks use layered architectures in which the network services are divided into different layers [ELLIS83]. Each layer in the hierarchy is a logical entity that performs certain functions. There is an inter-layer interface between
each pair of adjacent layers. Each layer accepts services from the layer immediately below it and provides services to the layer above it through these interfaces. As long as the interface between any pair of layers is clearly defined, each layer can be designed and implemented separately.

In an effort to standardize network architecture and protocols, the ISO (International Standard Organization) has developed a reference model for computer network layered architecture, namely the Open System Interconnection (OSI) model. Its structure is shown in Figure 2.2. The dotted lines from layer n (n>1) of one side to layer n of the other shows a virtual communication channel between them. The real communication between these two layers of the same level is carried out by passing data and control information to the layer immediately below it, until the lowest layer (layer 1) is reached. At the lowest layer, there is a physical communication channel between the two systems, as shown by the solid line in Figure 2.2. This physical channel carries the actual data communication.

The physical layer provides the mechanical and electrical connections among the nodes in the network. This layer is concerned with transmitting raw data, zeros and ones, over a communication channel.

The data link layer uses the service of raw data transmission provided by the physical layer and transforms between the raw data and meaningful data frames. It creates and detects frame boundaries and synchronizes frame transmissions. It may also detect and correct transmission errors.

The network layer controls the operation of the communication network. Among other things, the network layer determines the switching
method between communication links and the packets routing method within the network. Network layer may not exist in a local area network that employs a single communication link.

These three lowest layers provide the service of correct and reliable frame communication to the upper layers of the network. They determine the communication characteristics of the computer network.

The transport layer is the interface between the data communication service provided by the three lower layers and the upper three layers. The basic function of the transport layer is to control and monitor a reliable end-to-end transmission between the source node and the destination node. At the source node, it accepts messages from the session layer, splits them up into smaller units if needed, and passes them to the network layer for transmission. At the destination node, the transport layer accepts data from the network layer, assembles the message, if necessary, and passes it to the upper layer. This layer may also be responsible for end-to-end acknowledgements.

Following the transport layer is the session layer which is the user's interface into the network. A connection between two computer processes is usually called a session. The function of this layer is mainly to build up, monitor and disconnect communication sessions.

The presentation Layer deals mainly with the syntax of data in the transmission. It is concerned with such problems as presentation or data type. Text compression, encryption, and data code conversion are also covered in this layer.
The application layer is the highest layer in this ISO-OSI model. Its function depends on specific application program and computer resources. Any application that needs to exchange information with a process in a different node may call upon the lower layer functions for network communication.

The Educational LAN designed in this project uses this layered structure. The three lowest layers are implemented in a Network Interface Unit which is based on an Intel 8051 single board computer. The high level functions designed in this project use the services provided by these low layers to control and monitor the transmission.

2.6 Serial Data Transmission Protocols

The data communication in this EDLAN involves two parts. The first part is the transmission between the host computer and the Node Interface Unit. This is through the COM1 port in an IBM-PC and a serial port in the Network Interface Unit using the RS-232 serial communication protocol. The second part is the major communication channel among all the nodes in the network. It is an RS-485 bus that connects all the Node Interface Units in the network.

**RS-232 protocol:**

The communication between the IBM-PC and the Network Interface Unit is governed by RS-232 protocol. It is an asynchronous serial transmission protocol. Each character is transmitted with a start bit, five to eight data bits and one to two stop bits. The baud rate, parity characteristics, number of data bits and stop bits are programmable at both
the IBM-PC and the Network Interface Unit. In the EDLAN, the communication parameters are set to 9600 baud, no parity, eight data bits and one stop bit.

**RS-485 bus protocol:**

The **RS-485** bus is also an asynchronous serial bus. It uses differential coding and can operates at a higher data rates and longer distance than the **RS-232** serial bus. The global bus in **EDLAN** uses this serial communication protocol. It operates at 9600 baud-rate. The maximum length of the bus is 4000 feet.

The **EDLAN** global bus works in a multi-drop half-duplex operation mode. All the nodes in the network are tied to the same bus. At any time, only one node may use the bus to transmit. The **CSMA/CD** protocol is used to control the access of the bus.
a) Fully Connected Network

b) General Mesh Network

c) Ring Network

d) Bus Network

Figure 2.1 Computer Network Topologies
Figure 2.2 ISO OSI Layered Structure
CHAPTER 3

NETWORK INTERFACE UNIT AND COMMUNICATION PROTOCOL

3.1 Overview

The Network Interface Units (NIU) are MCS-51 family single board computers. They supervise the global bus and control the communications among the nodes in the network.

At the source node, the NIU accepts command and data packets from the host computer and transmits the packet to the destination node through the global bus. A message will be sent back to the host computer to notify whether the transmission is successful or not.

At the destination node, the NIU detects and receives the message from the bus. It may buffer the message in the on board memory or send the data to the host computer according to the control command in the message packet.

3.2 Network Interface Unit

The block diagram of a NIU is shown in Figure 3.1. It has a central processing unit, program and data memories, interface circuits to the RS-485 bus and to the host computer. It also has parallel peripheral ports to interface with a printer and to read the node number set by a dip-switch.

3.2.1 Central Processing Unit

The heart of the controller is an intel MCS-51 family microprocessor. It is an 8 bit microprocessor with one on-chip serial port
and one 8 bit parallel port. It can support up to 64k program memory space and 64k data memory space [INTEL83]. At the present time, both the program memory and data memory are combined into a single memory space. One 8k byte EPROM chip (2764) is installed at the address from 0 through 1FFFH. Three 8k byte RAM chips (6264) are installed at the address 2000H through 7FFFH.

The on-chip serial port is dedicated to the communication with other nodes in the network through the RS-485 bus.

3.2.2 Interface with Host Computers

As shown in Figure 3.1, the NIU interfaces with the host computer using RS-232 communication protocol through channel A of the on board 82530 dual port serial communication controller. Memory mapped I/O techniques are used for this serial port. It is located at the data memory address space C000H through C003H and configured to operate at 9600 baud, 8 data bits, one stop bit and no parity checking. A line driver (1488) and a line receiver (1489) are used to convert between the 82530 TTL signal levels and RS-232 levels of the IBM-PC serial port.

Serial port B of the 82530 dual channel serial communication controller is not utilized in EDLAN. It is reserved for future expansion. It can be used to communicate with another device that supports serial communication.

3.2.3 Node Address Setting and Printer Port

There is a programmable peripheral interface controller 8255 residing at the address space F000H through F003H.
Port A of the 8255 peripheral interface controller is used to read the node number that is set by a dip-switch connected to the port. This number setting should be unique for each node in the network. At present time, a maximum of 64 nodes can be supported in the network. Six bits of address are thus needed. The two most significant bits are tied to ground and ignored.

This switch can be set at run time to select node numbers dynamically. The instructor’s node number in this EDLAN is always set to one and is transparent to the users. Number 0 is reserved for a broadcasting address.

Port B and port C of the 8255 peripheral controller are dedicated to the interface of a printer. The eight lines in port B are used as data lines and the three lines from port C are used for handshaking signals.

3.2.4 Global Bus Interface Circuits

The circuits that control the communication through the RS-485 bus consist of a serial port, differential line driver and line receiver, carrier sensing circuit and collision detection circuit. The schematic configuration is shown in Figure 3.2.

The serial communication uses the Intel 8051 on-chip serial port. It is programmed to transmit data at a baud rate of 9600, one stop bit and eight data bits without parity checking.

A differential line driver (75174) and a differential line receiver (75175) are used to convert the signal levels between the RS-485 global bus and the 8051 serial port.
The carrier sensing is based on the operation of a monostable chip (74LS123). The chip is triggered by the one-to-zero transition of the digital signal on the global bus. After each triggering, the monostable will have an output of one with a duration of 15 ms. The output of the monostable then returns to zero, its stable state. If there is a data stream transmitted on the bus, the monostable is triggered at least once every character since there is at least one one-to-zero transition in the start bit of each character. The time between the triggering signal is at most 1.15 ms. (11/9600 sec.) This continuous triggering will guarantee a continuous one at the output of the monostable chip. If there is no data transmission on the bus, the monostable chip will return to its normal state zero because of the lack of the triggering signal.

The collision detection is performed by comparing the transmitted data with the received data. A mismatch between the two indicates a collision. The node involved in the collision will terminate the transmission and restart the message transmission at a later time.

3.3 Communication Data Format

The data transmission among the nodes in the network and between a host computer and the corresponding NIU at one node is in the form of data packets. Two kinds of packets can be found in the EDLAN for command and data transmission [HERZO86].
3.3.1 Control Command Packet

A control command packet is transmitted from the host computer in a node to its Network Interface Unit to activate an operation task. It contains seven fields as shown in Figure 3.3.

The meaning of each field is shown below:

**SF :** Start Flag; A single character " { " is used to indicate the start of the packet.

**NN :** Device Number; This field contains two hexadecimal characters in the range of 00H to 3FH which indicate the destination of the control packet. NN is set to 00H in the command packet for local NIU.

**P :** Prefix; This single character informs the addressed device of the way to execute the command. Three kinds of prefix command are used

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>Queued Task; The task is put at the end of Task Queue. It is activated at the conclusion of the execution of tasks preceding it on the task queue.</td>
</tr>
<tr>
<td>!</td>
<td>Immediate Task; The task is to be executed immediately.</td>
</tr>
<tr>
<td>?</td>
<td>Synchronized Task; It works similar to queued task except the task can be activated only when a synchronizing signal &quot;$&quot; is received.</td>
</tr>
</tbody>
</table>

**TT :** Task Number; This is a two hexadecimal character operation code which specifies the task to be performed.
Q: Postfix Control Character; It indicates how many times the task will be executed. If it is ".", the task is executed only once. If it is a "+", then after execution, the task is put back into the task queue for re-execution.

DP: Data Parameter; This optional field allows 0 to 5 pairs of hexadecimal character for use in executing a task. The existence of this field and the meaning of the data in this field is command dependent.

EF: End Flag; "]" denotes the end of a control command.

For an example, the command packet

\[00!2A.\]

will cause the local NIU to check how many mail messages have been received in the NIU mailbox and to send this number back to the host computer.

3.3.2 Message Packet

Message packets are used to carry data messages among network nodes. A message packet consists of six fields. Its format is shown in Figure 3.4. The meaning of each field is explained as follows:

SD: Start Delimiter; A single character "[" is used to indicate the start of a data packet.

DA: Destination Address; This field contains two Hexadecimal characters in the range of 00H to 3FH.

SA: Source Address; This field also consists of two hexadecimal characters ranging from 00H to 3FH.

CC: Control Command; This field contains a control command to
direct the destination node interface to perform certain operations. Its format is exactly as the Command Control packet shown in Figure 3.3 except that the Device Number field is omitted. It is specified by the DA field instead.

**MESSAGE**: Message Field; This field contains the message to be transmitted. It is of variable length with a maximum of 8k bytes and limited by the size of the available space in the on-board memory. It is always followed by the End Of File (EOF) indicator: "cntrl-Z".

**ED**: End Delimiter; A single character ""]" is used to denote the end of packet. The appearance of the character ""]" within the message field will cause no problem since the character ""]" will be taken as an ED only when it follows the EOF indicator.

For an example, if node 3 wants to send a message to node 7, the data packet is as follows:

```
[0703{:26.} Messages EOF]
```

The control command {:26.} directs the **NIU** at node 7 to send the transmitted message to the host computer attached to it.

### 3.4 Summaries

An **EDLAN** node consists of an IBM-PC as the host computer and a network interface unit. The software is also divided into two parts. The low level program is written in 8051 assembly language and resides in the program memory of 8051 based network interface unit. It implements the physical and data link layer functions in the network [THYE88, HERZO86].
The high level program provides the application functions and high layer communication functions in the ISO OSI model. It is developed using a \texttt{Pascal} program in host computers.

The two parts communicate through an \texttt{RS-232} serial port using the packet formats described in the previous section. As long as the communication protocol is exactly followed, each parts can be designed independently. Modifications on one part will generally have no effect on the other.

As an example, we may change the IBM-PC to another type of computer or change the software from \texttt{Pascal} to \texttt{C} in the host computer. The only requirement is that the new computer and the new software must communicate with the \texttt{NIU} according to the protocol described in the previous sections.

On the other hand, \texttt{NIU} can be changed without affecting the host computer. Other microprocessors can be used as the \texttt{CPU} and other medium access protocols like Token Bus protocol can be used instead of \texttt{CSMA/CD}. Even the network topology can be changed to a ring with the network interface circuitry re-designed.
Figure 3.1 Structure of Network Interface Unit
Figure 3.2 RS-485 Interface Circuit Diagram
Figure 3.3 Control Command Packet Format

Figure 3.4 Data Packet Format
CHAPTER 4

IMPLEMENTATION OF THE EDLAN SOFTWARE

4.1 Overview

The EDLAN application program consists of four functional groups.

The first group includes the communication functions which control the interface between the host computer system and the Network Interface Unit. It also includes the transport and session layer functions that start, terminate and control the message transmission among the nodes in the network. This group of functions are transparent to the users of EDLAN.

The second group consists of the housekeeping functions including file management functions and a built-in screen editor. Users can manage their host computer files or edit files without leaving the EDLAN software.

The third group involves CLASS and EXAM operations. They supervise the exchanging of class materials, exam materials, test results, questions and answers between the instructor and students. Functions are also designed to help the instructor analyze test results.

All the other functions fall into the last group. They have miscellaneous functions including menu selection functions, window functions, on-line help functions and reset functions.

The flowcharts in Figure 4.1 and 4.2 show the logical flow of the instructor program and student program, respectively. The users interface with the software through menu selection and window functions. They activate housekeeping, CLASS/EXAM, or other functions through their
keyboard. These functions may need to call communication procedures to finish their tasks. The relationship among these four groups of functions is shown in Figure 4.3.

4.2 Communication Functions

A group of functions are developed to control the interface between the IBM-PC computer and the Network Interface Unit and supervise the communication among all the nodes in the network. They are used by both instructor and student program.

4.2.1 Serial Data Buffer

A large FIFO buffer Com1 buffer of 4096 bytes is implemented in the main memory of the host computer to temporarily hold the incoming messages. Its structure is shown in Figure 4.4. The locations in the buffer are arranged in a logical ring structure with the last location followed by the first one. The pointer Buffer_head points to the first valid character to read from the buffer and Buffer_tail points to the first available location for the next character received from the serial port.

As one node may send to other nodes at any arbitrary time, the destination node is not able to determine when a message will come to its serial port. An interrupt driven technique is used to receive the asynchronous message. Whenever a character is received from the serial port, the main program is interrupted and an interrupt service routine is activated to transfer the character to the tail of the buffer. The main program may continue its job upon returning from the interrupt routine. The main program periodically checks to see if there is any message
received in the buffer. It reads out the message from the buffer and
determines the type of the message. Appropriate functions are called to
process the received message.

4.2.2. Serial Data Transfer Procedures

This group includes the following procedures:

1) Procedure Initialize_com1: It uses DOS interrupt function 14H
and the declared psuedo-register variables to initialize serial port 1 to 9600
baud rate, 1 stop bit and 8 data bit without parity[DUNCA86]. The
structure of the psuedo-register variables is as follows:

\[
\text{Registers} = \begin{array}{l}
\text{Record} \\
\text{Case integer of} \\
1: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags:Integer); \\
2: (AL,AH,BL,BH,CL,CH,DL,DH : Byte)
\end{array}
\]

2) Procedure RS232 interrupt: This is the interrupt service routine
that is activated by the arrival of a character at the serial port COM1. The
routine reads the received character from COM1 port and puts it in
Com1_buffer[Buffer_tail]. Buffer_tail is then incremented by one. Inline
machine codes are used to save all the registers in the interrupt routine.

3) Procedures Start_com1_interrupt and Stop_com1_interrupt: Those
are used to enable and disable the COM1 port interrupt. In the
Start_com1_interrupt routine, the serial port COM1 interrupt vector is
initialized to the entry point of the interrupt service routine.

4) Procedure Check_buffer: This procedure is called periodically to
check if there is any message received in Com1_buffer. When Buffer_head
does not equal to Buffer_tail, there are some messages in the buffer. Each message is always in the format as shown in Figure 4.5.

The first line of the message contains the source node number of the message. It is recorded for later use. For an example, when a test result is received by the instructor, the node number is used to determine from which student the test result comes.

The message type, appearing in the second line of the received message, is an integer number that is transmitted by the source node of the message. The messages transmitted from the students to the instructor have the following types:

Type 1: Test results for short answer problems;
Type 2: Test results for multiple choice questions.
Type 3: Questions concerning the class or exam materials.

In the opposite direction, the instructor sends the following types of messages to the students:

Type 1: Short answer exam problems.
Type 2: Multiple choice exam problems.
Type 3: True/False exercise problems.
Type 4: Multiple choice exercise problems.
Type 5: Class materials that will be displayed directly on the student’s screen.
Type 6: Class materials that will be saved on the student’s mass storage memory and displayed on the student’s screen.
Type 7: Messages other than class material. They are displayed on the student’s screen. The messages can be entered from keyboard or read out from a disk file.

Different operations are carried out to process messages of different types. For example, when a problem file is received by a student. The problems are saved on the student's disk and a short message is displayed to prompt the student to push <F1> to load the message.

The flowchart for the procedures Check buffer in the instructor's program and student's program are shown in Figure 4.6 and 4.7, respectively.

Besides the message types listed above, the instructor sends greetings and login messages to the students at the beginning of a class or an exam,. In this message, the instructor informs the students whether it is a regular class or an exam by transmitting an operation type in the second line of the message. Type 1 is used to indicate a class and Type 2 for an exam.

5) Procedures Build command packet and Send command packet: These two procedures set up a communication session. The first procedure builds a communication control packet which includes a start delimiter, control field, source address, destination address and end delimiter. The second one is used to transmit the command packet to the Network Interface Unit and then to other nodes. If the command packet is for transmitting a message, the message will be transmitted immediately following the command packet as shown in Chapter 3.
6) Procedures \texttt{Transmit\_string} and \texttt{Transmit\_file}: The first procedure transmits a string to the Network Interface Unit. It is built on the top of procedures \texttt{Build\_command\_packet} and \texttt{Send\_command\_packet} and calls the last two procedures to perform the required function. The second one calls the first to transmit a file to the Network Interface Unit.

7) Mail manipulation procedures: A mailbox is implemented in the \texttt{RAM} space of the Network Interface Unit. The low level control functions of the mailbox operation were written by Thye [THYE88]. The high layer functions use the following procedures or functions to access the mailbox:

Function \texttt{Number\_of\_packet}: This function sends a command packet to the Network Interface Unit to check how many mail messages have been received in the mailbox.

Procedure \texttt{Purge\_mail}: This procedure purges all the mails in the mailbox.

A specific mail is read out from the mailbox by sending the command packet

\begin{verbatim}
{00!26.HL}
\end{verbatim}

to the Network Interface Unit. In this packet, \texttt{H} and \texttt{L} are the higher byte and lower byte of the number of the mail to be read from the mailbox.

This set of communication control procedures provide the rest of the program an interface to communicate with the Network Interface Unit and perform the network functions.
4.3 Housekeeping Functions

The housekeeping utilities consist of file and directory management functions and a screen editor.

4.3.1 File and Directory Operation Functions

The file and directory management functions consist of the following procedures:

- Change Directory: Change the default directory to another directory.
- List Files: List the contents of files on the system screen.
- Remove Directory: Remove an existing directory.
- Make Directory: Make a new directory on the computer disk.
- Erase File: Erase a file from the disk.
- Rename File: Rename a file to a new file name.

All these functions except the last one were implemented by calling the corresponding standard Pascal library functions [BORLA86]. This library does not support functions to list a directory. It is implemented by using the DOS function calls provided by IBM-PC DOS operating systems [DUNCA86]. The pseudo-register bank variable shown in Section 4.2.2 is used to interface with the DOS function calls.

DOS function 1AH [DUNCA86] is first called to set a small memory area to hold the results of directory search function. The data structure is as follows:
Byte 0-20 reserved
Byte 21 attribute of the file
Byte 22-23 file time
Byte 24-25 file data
Byte 26-27 least significant word of file size
Byte 28-29 most significant word of file size
Byte 30-42 file name and extension

Then **DOS function 4EH** is called with search string "*.*" to find the first file in the directory. After that **DOS function 4FH** is called many times to search the rest of the files until all the files in that directory are found. Each time a file is found, its name and attributes are put into the memory area set by the previous **DOS function 1AH**. The data is then read out, processed and displayed on the system screen.

### 4.3.2 Screen Editor

The procedure **Editor**, a built-in screen editor, provides the instructor and students with a helpful tool to edit class materials, exam problems and other messages. The screen editor supports different window sizes. The users can edit text within the window defined by the calling program.

The editor works on the memory space that directly maps to the system screen. For a monochrome monitor, the memory area starts from location B0000H. It starts from location B8000H for color monitors [SARGE86]. The memory space of 4000 bytes serves as the image of the system screen with a size of 80*25 characters. Each character in the screen corresponds to one word in the memory. The lower byte of the word contains the ASCII code of the character and the high byte contains the
attribute that determines the color or brightness of the character and the background.

Assume the coordinate of the upper-left corner of the edit window is \( <x_1, y_1> \). The width and height of the window are \( w \) and \( h \), respectively. Let

\[
\text{Base} = \begin{cases} 
B0000 & \text{For monochrome monitor} \\
B8000 & \text{For color monitor} 
\end{cases}
\]

And assume that a character \( C_i \) is \( x \)-th character in \( y \)-th line of the edit window. Then the address of the memory location that maps to the character \( C_i \) can be found to be

\[
\text{Addr}(C_i) = \text{Base} + 2 \times 80 \times (y + y_1 - 2) + 2 \times (x + x_1 - 2) \\
(0 < x < w + 1, 0 < y < h + 1)
\]

In order to edit files whose sizes are larger than the window size, two large arrays are used to hold the text outside the screen. These are the array \( \text{FrontPage} \) and \( \text{BackPage} \). The text shifted out of the top of the edit window is saved in the first array. The text shifted out of the bottom line of the window is saved in the second array. Each array has a pointer that points to the last line shifted into the array. The maximum size of the file that can be edited is determined by these two arrays and the editor window size. At the present time, each array is declared to have a size of 100 lines. Large files can be accommodated by simply changing the declaration of these arrays.
The editor procedure continuously checks the keyboard to see if there is any key being pressed. The keys are divided into three groups: cursor movement keys, control function keys and text character keys. The cursor movement key group includes the four arrow keys. They are used to control the cursor movement on the screen. A pair of numbers \(<x,y>\) is used to keep track of the current cursor position. They are updated each time an arrow key is pressed or a text character is inserted or deleted.

The control function keys include:

\(<F8>\): Save; When \(<F8>\) is pressed, the current text file under editing is saved onto the disk. It includes the text on the screen, in the FrontPage and BackPage. Only the lower byte, the ASCII code, for each character is saved and the higher byte, the attribute, is discarded.

\(<\text{ESC}>\): Exit; The \(<\text{ESC}>\) key is used to exit from the edit routine. Files should be saved before this key is pressed.

\(<\text{Ins}>\): Toggle insert/overwrite mode; The screen edit can work at either of the two modes, the insert mode and overwrite mode, which is toggled by the \(<\text{Ins}>\) key.

\(<\text{Del}>\) : Delete the character under cursor.

\(<\text{Ctrl}>'Y\) : Delete the current line.

All the text character keys pressed are put into the memory space of the current cursor location of the screen image.
4.4 Class and Exam Functions

The major groups of utilities provided by the EDLAN software are the CLASS and EXAM functions. The instructor uses this group of functions to supervise classes and exams. The students use this group of functions to take an exam or a class.

4.4.1 Open Class and Exam Functions

The open class and open exam functions set up an environment for a class or an exam. In the instructor's node, this includes the following:

1) Prompt and get the class title.
2) Prompt and get the time length of the class or exam.
3) Prompt and get the number of students in the class.
4) Call the communication functions to transmit the class title and time length of the class or exam to the students. The transmitted message should also inform the students whether it is a regular class or an exam and prompt the students to send their names.

Upon receiving the class title, the student's program prompts the student to input his name and social security number to log onto the network class. It then calls the communication functions to transfer these two pieces of information to the instructor's mailbox. The program will automatically transfer to Class or Exam mode according to the type of operation given in the instructor's message.

After transmitting the class/exam title, the instructor's Network Interface Unit starts to receive student names from its mailbox. A record is set for each student in the class with the following format:
name: the student's name, maximum of 20 characters;
stu_SSN: the student's social security number;
node_num: the node number for the students station;
result: array of strings used to hold the student test result;
score: array of integers to hold the score for each problem;
graded: array of tags indicating if each problem has been graded or not;
total: the total score of the student;
final_grade: the final letter grade of the student;
get_result: a tag indicating if the student’s result has been received;
get_score: a tag indicating total score has been computed.
get_grade: a tag indicating the final letter grade has been assigned.
get_name: a tag indicating the student has been logged in the class.
receive_time: the time that the result has been received;

The program exits from the open class/exam function when the expected number of students have transmitted their names to the instructor or when the instructor hits the <ESC> key.
4.4.2 InClass and InExam Functions

The InClass and InExam functions are developed to supervise classes and exams. Their main body is a loop which continuously checks the serial buffer and system keyboard. When a key is pressed by the user or a message is received in the serial buffer, appropriate functions are performed according to the key pressed or message received. The flowchart for the InClass operation in the instructor's program is shown in Figure 4.8. The InExam procedure has a very similar structure. The counterpart in the student's program is shown in Figure 4.9.

The operation of checking the serial buffer is performed by calling the Check_buffer procedure. Whenever a message is received from the serial buffer, its node number is parsed out and recorded. The type of the message is also decoded and appropriate procedures are called to perform the required functions.

For the instructor's program, the following actions may be taken according to the type of the message received.

**Type 1:** Test results for general problems; When a data packet of this type is received, the instructor's program checks the source node number. A result file is opened for the result received from each source node. If an result file already exists for the same source node, the new result will overwrite the old one.

**Type 2:** Test results for multiple choice questions; The received result is saved on the disk. The procedure Machine_grading is then called to grade the received results. The implementation of the Machine_grading
subroutine is quite straightforward. It simply compares the received result file with the answer key file, problem by problem.

**Type 3: Questions concerning the class or exam materials;** When the procedure `Ask_question` is activated at a student node, the screen editor is called for the student to edit questions. At the end of the edit session, communication subroutines transmit the questions from the student node to the instructor.

A FIFO question queue is implemented on the system disk of the instructor's node. Each question received from the students is saved in a separate file. Two pointers are used to indicate the first and the last questions in the queue. The FIFO can hold maximum of 10 question files. The eleventh question will overwrite the first one if it has not been answered.

Meanwhile, a question flag is set and the word "Question" will be displayed on the screen to tell the instructor there is at least one question from students. Since the instructor may not want to answer the question immediately after it is received, the question will not be displayed until the instructor hits a function key.

The instructor can suspend his teaching work and enter the answering question mode by pressing function key <F6>. The first question in the FIFO queue will then be displayed on the instructor's screen. The program then opens a window and prompts the instructor to enter the answer. Upon exiting from the answer edit session, the answers are transmitted to the student who asked the question.
The operations to be performed at the student side are listed below:

**Type 1,2:** Short answer or multiple choice exam problems; The problems received from the instructor will be saved in a disk file. The students can then load the problems into memory and start to solve them. These two kinds of problems are separated since the answer to a type 2 problem is always a single character. However, the answer to a type 1 problem may have more than one line. Different subroutines are called for the students to solve these two kinds of problems.

**Type 3,4:** True/False or multiple choice exercises problems; These two types of messages contain both the problem and the answer keys. These two fields are separated by a line starting with a single 0. The problems and the answers will be loaded into the memory separately. These two types of problems are separated because of the difference in the problem solving operation. For the True/False problems, there are only two possible answers. If an incorrect answer is entered from the keyboard, the program simply displays a message to indicate it is not correct. The correct answer is then always obvious. However, for the multiple choice problems, there are more than two possible answers. If an incorrect answer is entered, the program displays a message and prompts the student to reenter the answer. The correct answer is displayed if the second answer is not correct either.

**Type 5:** Class materials; This type of material is displayed directly on the student’s screen.

**Type 6:** Class materials; This type of class materials will be saved on the student’s disk; The message will also be displayed onto the student’s screen.
**Type 7: Messages**; The messages received from the instructor are directly displayed in the appropriate window on the screen.

Operations can also be activated by hitting a function key. The InClass/InExam functions continuously check the keyboard input to determine a function key or <ESC> key has been pressed. They are detected by the receipt of the following scan code from the keyboard buffer:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>&lt;ESC&gt; key</td>
</tr>
<tr>
<td>27,59</td>
<td>F1;</td>
</tr>
<tr>
<td>27,60</td>
<td>F2;</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>27,67</td>
<td>F10;</td>
</tr>
</tbody>
</table>

The following procedures are activated by function keys.

1) Procedure Help; It displays help text to the system screen. The design and application of the On-line help functions will be explained in Section 4.4.

2) Procedure Load problems; A memory space of 120 lines are reserved to hold problems during the problem solving session. The procedure Load problems is activated in the following cases:

   a) When a student starts to solve exam problems in an exam.

   b) When a student starts to solve exercise problems in a class.

   When this procedure is activated, the problems are brought into the reserved memory space from disk files. At the same time, the problems are
also displayed on the computer screen for a brief review. The students can then work on the problems residing in the memory.

3) Procedures that manipulate problems and answers; After loading problems into memory, the instructor and students can call appropriate functions to solve the problems or setup answer keys.

A test or exercise generally consists of several problems. Each problem may have a different number of lines. As in the real case, a problem is identified by a problem number at the beginning of the first line of the problem. In EDLAN software, a specific problem is selected by searching from the beginning of the problem array until the given number matches the number at the beginning of the problem line. All the texts starting from this line until the next line that has a decimal digit at the beginning are for the same problem. For this purpose, only the first line of a problem can start with a decimal digit to indicate a problem number.

To solve a problem, the problem number is first prompted from the system keyboard. The software is designed to print a default number before it reads the value from the keyboard. The default value is obtained by incrementing the number of the previously worked problem starting from problem number 1. If the user hits the <Enter> key without typing any number, the input will be read as a string with a length of zero. In this case the default value will be used. If a number is entered, the string length will not equal to zero. The string is then converted to an integer value and will be used as the problem number.

When a problem is selected, it is displayed in the upper window. The answers are to be entered from the system keyboard. If the problem is re-
selected next time, the existing answer is also displayed. This allows the users to review and correct answers if they want. If the user hit the single <Enter> key, a line with a length of zero is read in. The existing answers then remain unchanged. Otherwise, the new lines are used to overwrite the existing ones.

4.4.3 Grading Functions

There are two kinds of problems to grade. The first kind is multiple choice problems including True/False questions. This type of problems are graded by the computer automatically. For each set of problems, there is a corresponding answer-key file. During an exam session, the answer key is loaded into a memory array. When a test result is received from a student, the procedure Machine grading is called to compare the received result with the answer keys. The scores are computed, displayed and saved in the student record.

The second type of problems are short answer problems. They are saved in the computer disk and must be graded by the instructor. A flexible grading procedure, Set grade is developed to help the instructor to grade this kind of problems. It can be activated at any time by selecting "Grading" from the "Test" sub-menu. Set grade works in a way similar to the problem solving function. It prompts the instructor to enter the student number and number of problem to grade. Default values are also displayed. After each problem (each student) is finished, two options are given to the user. He can continue to work on the next problem (next student) or exit from the grading function. Since the student number and problem number can be selected using the keyboard, the user can always quit the operation and
return at a later time. Each time a problem is selected, the answer from the student is displayed in the upper part of the window. The scores are then prompted from the keyboard and will be displayed in the lower part of the screen.

As described in section 4.4.1, there are several fields in the student record concerning test grading functions. They are explained as follows:

**score** is an array of integers to hold the score for each problem. When problem i is graded, its score is saved in score[i].

**graded** is an array of tags indicating if each problem has been graded or not. graded[i] is set to True after problem i is graded. If graded[i] is True already when problem i is selected by the instructor to grade, the original value of score[i] is then displayed together with the student answer. The instructor can then enter a new score to overwrite the original one or hit <Enter> to keep the original score[i]. This allows the instructor to review and change the scores.

**total** is the total score for each student. It is computed when all the problems have been graded.

Two other fields are concerned with the final letter grade assignment functions. **final grade** is the final letter grade of the student and the field **get grade** is a tag indicating the final grade has been assigned.

When the procedure **Record** is called after all the problems of all the students has been graded, the minimum, maximum and average score of the student are computed. The procedure then prompts the user to enter minimum scores for grade A, B,C,D respectively. The letter grade of each
student is then computed automatically by the computer.

4.5 Miscellaneous Functions

4.5.1 On-line Help Function

An On-line help utility is developed for the users to get on-line help at any time. The help text is contained in a large constant array in data memory. A procedure is designed to display a section of help text to the help window. It is passed with two line numbers, the start line number and end line number, as parameters. In this way, one function can be used to display different help menus. These two line numbers are determined and passed to the function by the calling routine at the time the help function key is pressed.

4.5.2 System Reset Functions

The system reset functions initialize all the variables used in the program. The users can also use this group of functions to set the system clock and date. IBM-PC DOS functions are called for the data and time setting.

4.5.3 Menu Selection Functions

There two kinds of menu selection functions in the instructor's program, the main menu selection and sub-menu selection.

The procedure Main_menu_selection determines the selected item index from the main operation menu. The main body of this procedure is to check the keyboard input. Initially, the first menu item is highlighted and the first character of each command is in boldface. A global variable
main_menu_index is used to keep track of the index of the main menu item selected. It is initialized to one. Each time an arrow key is pressed or the first character of a command is pressed, the main_menu_index is updated accordingly. The function exits when <Enter> key is pressed or when the first character of any command is pressed. The returned value of main_menu_index is used to determine the functions to call.

The procedure Sub_menu_selection works in a way similar to the procedure Main_menu_selection does. There are two differences between them.

The first difference is that down and upper arrow keys are used instead of left or right keys since the menus are displayed vertically on the screen. The other difference lies in the fact that one procedure is used to select the sub-menu index for all the sub-menus. Since only one sub-menu list is currently used at any time, a single variable sub_menu_index is used to return the index of the menu item selected. The operations are different for different sub-menus even if the sub_menu_index is the same. They are determined by the calling subroutines.

A string consisting of the first character of all the current sub-menu items is passed to the procedure. The character typed from the keyboard will be sequentially compared with this string. The first match position will be the index of the menu item selected.
4.5.4 Window Management Functions

A window is an area on the computer screen that is encircled by double/single line borders. The following parameters determine the characteristics of the window:

a) The coordinate of the upper-left and lower-right corners of the window.

b) The background color, text color, border color of the window.

c) The type of border, single line or double line.

4.6 Summaries

The four groups of functions discussed in this chapter constitute the EDLAN high level software. They control and monitor the data communications among the instructor's node and student's nodes. They supervise and monitor the class and exam application operations. Window functions, menu selection functions and housekeeping functions are also developed to help the users of the network for the ease of use and management. The communication functions are controlled by EDLAN software and transparent to users. This transparency has two advantages:

1) Users need not know the technical detail to transmit their data.

2) It allows the instructor to select destination node but prevents students from doing so. All messages from student nodes are automatically transmitted to instructor's node. This is necessary for CLASS and EXAM operations.
Figure 4.1 Flowchart of the Instructor Main Program
Figure 4.2 Flowchart of Student Main Program
Figure 4.3 EDLAN Software Structure
Buffer_head: ---\rightarrow To where data is used

Buffer_tail: \leftarrow--- From Serial Port

Figure 4.4 Com1_buffer Structure

....... Packet received from NODE node number ...... message type

messages

Figure 4.5 Format of the Messages Transmitted from NIU to the Host Computer
Figure 4.6 Flowchart for Check_buffer Routine in Instructor Program
Figure 4.7 Flowchart of the Check_buffer Routine in Student Program
Figure 4.8 Flowchart of the InClass Procedure in Instructor Program
Figure 4.9 Flowchart of the Exam Operation in Student Program
CHAPTER 5
SUMMARIES AND SUGGESTIONS FOR FUTURE WORK

5.1 Results and Summaries

Previous chapters of this thesis have discussed the design and application of EDLAN, an EDucational Local Area Network. The primary goal of this project is to design a powerful, user friendly software that enables instructors and students to exchange class and exam materials using the network. This design goal is realized by a 5000 line instructor program and a 3000 line student program.

A network with two nodes has been built and tested. One node is used by the instructor and the other by a student. During the test, the node number on the NIU of student’s station is changed to different values to simulate a multi-student classroom environment. All the login session, class and exam session including question and answer sub-sessions and test grading sub-sessions are successfully tested. All the class and exam data are correctly recorded. The test result shows that the final design of the EDLAN software meets the design goal.

The software has the following characteristics:

1) It includes a set of high layer network interface functions that control the data communication among different nodes. The source node builds up a message packet and sets up a communication session to transmit the packet to its destination. The message packet is then identified, disassembled and passed to the application layer at the destination node.
2) It supports complete class and exam operation functions. This includes the student login functions, class material distribution and test problem distribution functions, question and answer functions, student result submission functions and grading functions.

3) It has helpful housekeeping functions to enable the instructor to manage the host computer system. This includes the file operation functions and a screen editor.

4) It uses a friendly menu driven mechanism. The users can activate any operation by simply hitting a command key.

5) Window operations make the software more attractive. Different messages are displayed in different windows. The windows may have different colors if a color graphic card is used. The type of graphic card is determined automatically by the program. There is no need to reconfigure program for different graphic cards.

5.2 Suggestions for Future Work

This thesis presents a preliminary design of the EDLAN. Many areas deserve further design and research. Some of them are listed below.

1) The high level functions at the student’s node can be moved from the host computer to the 8051 single board computer. The students can then use terminals rather than computers to perform the same functions. This will greatly reduce the total network cost.

2) A new microprocessor chip, the 8044, can be used to further improve the network performance.
3) Token bus can be used in the network as an alternative to the CSMA/CD protocol. Their results can be compared.

4) More powerful fault tolerant communication should be implemented. The network operation should recover easily from network hardware malfunctions and/or transmission errors.
BIBLIOGRAPHY


APPENDICES
APPENDIX A
USER'S MANUAL FOR INSTRUCTOR PROGRAM

A.1 Overview

The application program at the instructor side is called INSTR. It provides all the functions to supervise classes and exams, to analyze exam results and to manage file systems.

The program is designed to run on any IBM-PC/XT/AT or compatible computers with Hercules, CGA or EGA graphics card. The program will adapt to the graphic card automatically. It can run from a floppy disk or from a hard disk. However, the use of a hard disk will greatly speed up the system operation.

A.2 Start of the Program

To run this program, change to the directory which contains this program and type

INSTR <Enter>.

If the message "Node Not Ready" appears on the screen. The Network Interface Unit attached to this station may not work properly. Push the reset button on the attached NIU and then hit a key to continue this program.

As shown in Figure A.1, there are six groups of operations you can select from the main operation menu, namely File, Reset, Class, Test, Edit and Quit. These operations are explained in detail in the following sections. There are two ways you can select a command:
1) Use left or right arrow keys to highlight the command and then hit the <Enter> key.

2) Type in the first character (boldfaced) of the command.

For an example, typing "f" or "F" will cause the execution of the file operations. Moving the cursor to highlight "File" command and then hitting <Enter> key will cause the same operation.

For each of the first four operations (File, Reset, Class, Exam), a sub-menu window will be opened. You can select any command from the displayed sub-menu using the following two ways:

1) Use the up or down arrow keys to highlight the command and then hit the <Enter> key.

2) Type in the first character (boldfaced) of the command.

A.3 File Operations

File operations are used to manage the computer file system. Its sub-command menu is shown in Figure A.2 and explained in this section. Please refer to your DOS manual for a detailed description of file systems.

A.3.1 Show Directory

This command is used to list the file or subdirectory names in the current directory. A small window is opened to display the file and subdirectory names. In the window, "F" following a name is used to indicate a file and "D" for a subdirectory.
A.3.2 Make Directory

Make Directory command is used to make a new directory. You will be prompted to enter the new directory name.

A.3.3 Change Directory

Change Directory command is to change the default directory to a new directory. After you have entered the new directory name, the files and subdirectories in the new directory will be displayed.

A.3.4 Erase Directory

This command is used to erase or delete an existing directory. You will be asked to enter the directory name.

A.3.5 Delete File

This command is used to delete a file from the disk. The file name will be prompted from the computer keyboard.

A.3.6 Rename File

This command is used to change the name of a file to a new name. Both the old name and new name of the file should be entered from the keyboard.

A.3.7 List File

This command is used to list a file on the system screen. You will be asked to enter the file name from the computer keyboard.
A.3.8 Print File

This command is used to print a file on the system printer.

You can exit the file operation and go back to the main menu by pressing <Esc> key.

A.4 Reset Operations

The reset functions can be used to set the system time and date.

A.4.1 Set Date

The current system data is first displayed in the format of month-date-year, each field has two digits. The new date can be typed in the same format. A single <Enter> key will keep the current date unchanged.

A.4.2 Set Time

The current system time is displayed first in the format of hour:minute:second, each field has two digits. The new time can be typed in the same format. A single <Enter> key will keep the current time unchanged.

A.5 Class Operations

The class operation functions initiate, supervise and monitor classes. Its sub-menu window format is shown in Figure A.3. OpenClass is used to start the class. It should be activated before the other two. InClass should be selected after the opening of the class. It is the main body of the class functions. The Status command can be selected at any time to check the status of the class which include the number of the students who have
logged in and the name list of all these students.

A.5.1 Open Class:

The OpenClass operation initiates the class environment. When it is activated, a new window is opened and the following line will appear in that window:

Please enter the class title:

Type in the class title followed by <Enter>, computer will display:

Time of class in Min (50): 

Enter the class time length in minutes and then hit <Enter>, or just hit <Enter> to use the default value 50 minutes. Then the computer will prompt the expected number of student in the class by displaying the following line:

The expected number of students:

Simply enter the number of students in the class and then hit <Enter>. There is no default value for this question, a number must be entered. If the number of students of the class is not known at this moment, the number of nodes in the network can be used here. No more students than this number can log onto the network.

After the above information has been entered, a message which contains the class title and class time length will be automatically transmitted to all the students' terminal in the network. The program is then ready to receive students' log-in information.
A class status window as shown in Figure A.4 is displayed. Every
time a student logs in the network through his computer, his name and
social security number will be displayed in this status window. The
procedure is repeated until the number of students received equals to the
expected number of students of the class. You can press <Esc> key to exit
from the OpenClass operation before the expected number of students log
in.

The meaning of other fields in the status window is explained in the
following:

Num : An ordinal number for each student in the class;
Node : The node number of the computer which the
        student is using;
Finish:The time when the student submitted his test
        result. A "NO" in this field indicates that
        the student has not submitted his test
        result.
Score: Total test score of the student;
Grade: Final letter grade of the student.

The last three fields are useful only for exams. They are ignored in a
non-exam classes.

A.5.2 InClass:

This command is selected to teach a class. When it is activated, two
windows will be opened in the screen. They are shown in Figure A.5. The
upper window is for the messages received from students. The lower
window is to display the message that you have typed from the system keyboard. The top line of the screen shows the class title, the current date, the count-down timer for the class and the current time.

A list of commands that can be selected is shown at the bottom of the screen.

<F1>: Help; To pop up the on screen help menu for the class operations

<F2>: Answer questions; If one or more questions have been received from students, a message "Question" will be displayed on the screen. You can then press <F2> to answer these questions. When you press the <F2> key, a question from a student is then displayed and a window is opened for you to compose your answer. The answer is then sent back to the student who asked the question. If several questions have been received, they are answered in a First-In-First-Answer order.

If no question has been received and this key is pressed, a short message 'No Question' will be displayed on the screen.

<F3>: Send Message; This command is used to edit a message and transmit the message to all the students or to an individual student. After <F3> is pressed, the question "Send to all the students (Y/N) ?" is displayed on the screen. If "Y" is selected, the message will be sent to all the students. Otherwise the message will be sent to an individual student whose station number is prompted from the keyboard.

<F4>: Transmit File; This command has the same operation as the command "send message", except a file on the disk is sent instead a message
typed in from keyboard..

  <F5>: Exercise; This command will transmit class exercise problems to all the students. After pressing <F5> key, you can select one of the following two options:

  F : To transmit exercise problems from a file. The file name is prompted from the system keyboard.

  S : To edit exercise problems and then to transmit them to the students.

You need also to specify what kind of exercise problem (Multiple choice or True/False) you are sending. The problem type will be transmitted to the student together with the problems.

The answer keys are also transmitted to the student so that they can check their results. If the problem is to be edited on the screen, the computer will prompt you to enter the answer key for each problem. For the format of the problem files, please refer section A.7, Edit Operation, for detail.

  F6: Status; This command allows you to review the class status which includes the name, social security number, node number etc. The class status is displayed on the system screen.

  <Esc>: To quit the InClass operation and go back to the Class submenu.
A.5.3 Class Status:

The class status functions are used to display the status of the class. It is equivalent to the operation activated by pressing <F6> in the Inclass operation mode. For the detailed explanation of the status information, refer to section OpenClass operation.

A.6 Exam Operations

The Exam operation functions initiate, supervise and monitor the examinations. Its sub-menu window format is shown in Figure A.6. OpenExam is used to start the Exam. It should be activated before the other two. InExam should be selected after the opening of the Exam. It is the main body of the Exam functions. The Status command can be selected at any time to check the status of the class which includes the number of the students, the name list, exam information of each student.

A.6.1 Open Exam:

The Open Exam operation initiates the Exam environment. Its operation is the same as that of OpenClass. Please refer to section A.5.1 for detail.

A.6.2 InExam:

The InExam operation works in a similar way as the InClass operation. The windows displayed on the screen have the same meaning as that in the InClass operation. The top line of the screen for exam operation shows the class title, exam type (short answer or multiple choice), the time left for the exam and the current time. The Command keys are also
displayed at the bottom of the screen. They are listed in the following:

F1: Help; To pop up the on screen help menu for the exam operations

F2: Answer questions; To answer student’s question. The operation is the same as that of InClass.

F3: Send Message; This command allows you to send any messages to all the students or any individual student. When <F3> is pressed, the computer will ask you whether you want to get the message from a disk file or edit the message from the keyboard. If you select the file option, you need to specify the file name. Then the message will be read from the file and transmitted to the students. If you select to edit the message from the keyboard, a message window will be opened for you. You can edit your message and transmit to the student.

F4: Send Problem; This command is to transfer the exam problems to all the students in the network. The operation is the same as that of sending exercise problems in InClass operation except the following:

1) No answer keys are transmitted to student.

2) There are two kinds of problems, short answer problem and multiple choice (True/False) problems. You will be prompted to specify the type of the problem. The window and message format is shown in Figure A.7. The exam type will be displayed on the screen after you make your selection. The window format for the exam session is shown in Figure A.8

F5: Status; This command is to monitor the class status. Refer the section InClass for detail.
Finally, the <ESC> key is used to exit from InExam and return to Exam sub-menu.

A.6.3. Grading

This operation is selected to grade students' exam results. The following operations can be activated during grading:

F1: Help; To pop out the on-line help messages.

F2: Review Result; To list a student's exam results on the screen. The student number is entered from the keyboard. Hit <Enter> key for the default value displayed on the screen.

F3: Grading; The results for multiple choice type problems will be graded automatically by the computer program. The grading function here is used to grade the students' results on short answer problems. You can select which student's result to grade by entering the student number. For each student's results, you can select any problem to grade by entering a problem number. The student solution for the selected problem will be displayed on the screen and the score for this problem can be entered from the keyboard. Hit <Enter> key without typing any number to use the displayed default value.

If you want to exit from the grading operation, select "N" in response to both the questions "Next problem(Y/N)?" and "Next Students(Y/N)?"

F4: Record; To set final letter grades and display exam information. When F4 is selected, the maximum score, minimum score, average are automatically computed and displayed on the screen. The message "Set
Letter grade (Y/N)?" will be displayed. You can choose to set the letter grade. If this is the case, the minimum scores for grade A, B, C, D are entered from the keyboard, the final grade of each student is then determined automatically by the program. The grade information is then displayed on the screen.

F5: Print; To print the class exam results including the name, social security number, node number, total score and final grade of each student.

A.7 Edit Operation

A screen editor is used to edit exam problems and general files. When Editor is activated, three options are displayed.

a) General files
b) Exam problem files
c) Exercise problem files

The first option is general files (non exam/exercise problem files), the file name is prompted from the system keyboard. It can be a new file or an existing file. The file size is limited to 120 lines. If a type b) and type c) file is selected, you will be asked whether the problems are general short answer problems, multiple choice problems or True/False problems. For the last two kinds of problems, the answers will be entered after the problem edit session so that they can be graded automatically by computers.

During the edit session, you can use arrow keys to move the cursor on the screen. <Del> key is used to delete a character. <Ins> key toggles the insert and overwrite mode. <Ctrl-y> is used to delete one line.
At the end of the edit session, press "F8" to save the file and push <Esc> key to exit from the edit session.

Class b and c files may have variable number of problems. However, the following rules must be observed:

1) The number of problems in one file can not exceed 30.

2) The total number of lines in each file can not exceed 120.

3) Each problem may have variable number of lines. There must be a problem number at the beginning of the first line of the problem. The first character in the following lines cannot be a decimal digit.

An example of exam problem files are shown in Figure A.9.

If the problems are multiple choice or True/False type, you are prompted to enter the answer-keys after the edit session of the problem. Each problem will be displayed on the screen and the answer to that problem can be entered from the keyboard. This process starts from the first problem to the last problem.

A.8. Quit

Quit from the INSTR program.
Figure A.1 Main Operation Menu for Instructor Program

Figure A.2 File Operation Menu for Instructor Program
Figure A.3 Class Operation Menu for Instructor Program

Figure A.4 Class Status Window Format
Figure A.5  InClass Operation Window Format for Instructor Program
Figure A.6 Test(Exam) Operation Menu for Instructor Program

Figure A.7 Exam Type Selection Window Format
Figure A.8 InExam Operation Window Format for Instructor Program
GEOGRAPHY
Final Exam

1. Where is the capital of Oregon?
   a) Portland
   b) Salem
   c) Corvallis

2. What is the population of Corvallis?
   a) below one million
   b) between one million and two million

3. In which city is OSU located?
   a) Corvallis
   b) Eugene
   c) Salem

Figure A.9. Example of Problem Files
APPENDIX B
USER'S MANUAL FOR STUDENT PROGRAM

B.1 Overview

The application program at the student station is called STUDENT. It provides all the functions to log in the network class, to take a class or take an exam.

As with the instructor program, this program is designed to run on any IBM-PC/XT/AT or compatible computers with Hercules, CGA or EGA graphic card. The program will adapt to the graphic card automatically.

B.2 Start of the Program

To run program STUDENT, change to the directory which contains the program and type

STUDENT <Enter>,

The computer is ready to receive messages from the instructor and ready for you to log in and take a class or exam.

If the message "Node Not Ready" appears on the screen. The Network Interface Unit attached to this station may not work properly. Push the reset button and then hit any key to continue this program.

B.3 Login

At the beginning of each class or exam session, the instructor indicates whether it is a normal class session or an exam session, specifies the time length for the session and initiates the login procedure. During the
login procedure, the program prompts you to enter your name and social security number. An example of the screen display during the login procedure is shown in Figure B.1. Type in your name and social security number in response to the system prompt to finish the login session. The information you typed in will be transmitted to the instructor's computer automatically.

B.4 Taking a Class

If the instructor has indicated it is a class session, the program will automatically transfer to the class session after login procedure. The screen format for a class session is shown in Figure B.2. Three windows are opened on the computer screen. The upper right window displays the station number, the login name, time, date and count-down timer for the class session.

The upper left window contains the messages other than class materials from the instructor. Examples of this kind of message are greeting messages or answers to questions concerning the class material.

The lower window is the largest among the three and used to display class materials and exercise problems.

The operation commands that can be activated in the class session are displayed at the bottom line of the screen. Each operation is activated by press the corresponding function key.

<F1>: To load the exercise problem received from the instructor. If no problem has been received, the message "No problem received" will be displayed.
<F2>: To pop up the help menu for class operations.

<F3>: To start to solve exercise problems. Each problem is displayed on the screen. You can enter the solution to that problem as shown in Figure B.3. If the problem is multiple choice type, the solution will be compared with the answer key provided by the instructor. If there is a mismatch between the two, a message is displayed to prompt you to re-enter the answer. If the second answer is not correct either, the answer key is then displayed. For True/False type problems, computer will tell you either your answer is correct or not and then go on to the next problem. This procedure is repeated from the first problem until the last problem or a "No" is entered in response to the prompt "Next problem ?".

<F4>: To ask the instructor a question concerning class material or exercise problems. When this key is pressed, a small window is opened as shown in Figure B.4. The question can be typed in the window. Pressing <F8> will cause the question to be sent to the instructor and the window to be closed. The answer from the instructor will be displayed in the upper left window.

<ESC>: To quit the class session.

B.5. Taking an Exam

The display format for an exam is exactly the same as that for class except the command list on the bottom line of screen is different. The operation that can be activated during an exam session is listed below:

<F1>: Load exam problem. This step should be done before the problems can be solved. A message that shows whether it is a multiple
choice question problem, True/False problem or short answer problem is displayed upon the problem is received from the instructor. \(<F1>\) then can be pressed to load the problem. If no problem has been received when this key is pressed, an error message will be displayed to show that fact.

\(<F2>\) : To pop up help menu for the exam session.

\(<F3>\) : To start to solve problems. The operation is the same as exercise problem solving in a class session except the following:

1) No answer keys have been received for exam problems. Your answers will be saved onto disk and sent to the instructor instead of being compared with the answer keys for exercise problems.

2) The problem can be short answer problem type. In this case, the answer to a problem may have several lines.

3) You can select any problem to solve by entering a problem number in response to the computer prompt. Hit the \(<\text{Enter}>\) key without typing any number to use the given default. The default problem number is one greater than the number of the last problem solved. If you want to review or change the answer to a certain problem, type the problem number at the system prompt. Both the problem and your old solution will be displayed. Enter your new solution to overwrite the existing one. If you just hit \(<\text{Enter}>\) key, the existing solution will be kept.

\(<F4>\) : To ask questions concerning the exam materials. It works in exactly the same way as that in class session. Please refer to the last section for detail.
<F5> : Review exam problems and answers. The computer will display each problem and the answer to that problem starting from the first one through the last. Upon the old answer is displayed, you can hit the <Enter> key to review the answer to the next problem. If you want to change the answer, simply quit the review operation and go back to the solve problem session to change the answers.

<F6> : To transmit the answers to the instructor. This is the last step needed for an exam session.

<ESC> : To quit the exam session.

Three minutes before time-out for the exam, a message "three minutes left" will be displayed. You should check your solutions and prepare to submit them to the instructor. When the count-down timer reaches zero, the answers will be automatically sent to the instructor. A space will be sent for the unsolved problem.

B.6 Quit

To quit the program and log out the class, simply press <Esc> to exit the exam or class session and press <Esc> again.
** Message from Professor **

WELCOME TO GEOGRAPHY CLASS
Time Length in Minutes : 50

Enter your name (up to 20 char): John Bailey
Enter your Social Security Number: 555-55-5555

Use <BackSpace> to correct errors when done

Figure B.1 Window Format for Student Login Session
The capital of Oregon is Salem.
The largest city in Oregon is Portland.
Oregon State University is located in Corvallis.

Figure B.2 Window Format for Student Class Session

1) What is the capital of Oregon?
A) Portland
B) Salem
C) Corvallis
D) Eugene

Enter your answer: A
Message from professor

Good Morning Everyone.

Station : 6
Name : John Bailey
Time : 10:02:30
Date : 06-30-88
Time Left : 17:30

Class material

Enter Question here

What is the population of Portland ?

Figure B.4 Window Format for Student Question Session