

i-UTP Building Utilities Control using GSM - SMS Services

by

Noriza bt Zakaria

Dissertation submitted in partial fulfilment of
the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

DECEMBER 2004

Universiti Teknologi Petronas

Bandar Seri Iskandar

31750 Tronoh

Perak Darul Ridzuan

t

TK

6570

.M6

N841

2004

- 1) Mobile communication systems
- 2) EE - Thesis

CERTIFICATION OF APPROVAL

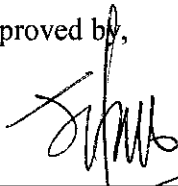
i-UTP Building Utilities Control using GSM - SMS Services

by

Noriza bt Zakaria

A project dissertation submitted to the
Electrical & Electronics Programme
Universiti Teknologi PETRONAS
in partial fulfilment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,



(Mr Mohd Azman Zakariya)

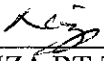
UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

December 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



NORIZA BT ZAKARIA
820909-02-5036
Matric ID:1642

ABSTRACT

Telecommunication system is one of the major components in industry. It can be summarized as the transmission, reception, and processing of information between two or more locations, using either digital or analogue transmission. The rapidly growing Global System for Mobile Communication (GSM) industry has provided the need for further studies on its capabilities and producing more useful services. This project will combine the Short Messaging Service (SMS) with the Peripheral Interface Controller (PIC) capability to produce another system that will be able to monitor and control the utilities in a certain building such as the air conditioner, lamps and doors. This system will make building monitoring and controlling a much easier task for the maintenance personnel. This system is also equipped with a security features to ensure that this system will not be misused by other unauthorized people.

The platform that this project has built offers a wide variety of other new system. There are several projects that are being conducted using the same platform or principal as this project which is SMS based system. Among the topics is SMS Car Parking Payment, SMS Bill Reminder Payment Systems, Machine-to-Machine (M2M); Mobile-to-machine and Machine-to-mobile, Home Surveillance with Mobile Phones and Mobile Phone Based Ticketing (transportation, train, parking meters etc). It is hoped that this system will gives benefits to the community and can be implemented in real environment in the future.

ACKNOWLEDGEMENT

I am very grateful to Universiti Teknologi Petronas for giving me a golden opportunity to conduct my research and use the facilities and equipment provided in the lab. My supervisor Encik Mohd Azman b Zakariya should be greatly acknowledged for his supervision, guidance and the project plans that he had prepared for me. I also would like to thanks all lecturers and staff who have been very helpful in assisting me either directly or indirectly in completing this thesis.

My special tribute to Ms Siti Hawa, the lab technician for final year project for helping me with the project. My special thanks also to my closest friend Azizan Hashim, Wan Zaharah Mohd Nazar and other students of the Universiti Teknologi of PETRONAS for their encouragements, advises and the priceless feedbacks which drive and motivate me and bring the best out off of me while completing tasks and works. Their support, comment and cooperation are much appreciated.

Finally, I am forever indebted to my family for their support and encouragement when it was most required. A final word of thanks to God for making this project successful and can be completed in the time given.

TABLE OF CONTENTS

CERTIFICATION OF APPROVAL.....	i
CERTIFICATION OF ORIGINALITY.....	ii
ABSTRACT.....	iii
ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	vii
ABBREVIATIONS AND NOMENCLATURES.....	viii
CHAPTER 1: INTRODUCTION	1
1.1 Background of Study.....	1
1.2 Problem Statement.....	2
1.3 Objectives and Scope of Study.....	3
1.3.1 Objectives of the Project.....	3
1.3.2 Scope of Study.....	4
CHAPTER 2: THEORY	5
2.1 Global System for Mobile Communication (GSM).....	5
2.1.1 History of GSM.....	5
2.1.2 Services Provided by GSM.....	6
2.1.3 Architecture of the GSM Network.....	9
2.1.4 Radio Link Aspect	12
2.1.5 Network Aspect	13
2.2 Peripheral Interface Controller (PIC)	16
2.3 Serial Interface	17
2.4 Overview of the System.....	19
2.5 Nokia 3310 and F-Bus Protocol.....	21
2.6 Nokia F-Bus Protocol Characteristics.....	22

CHAPTER 3:	METHODOLOGY/PROJECT WORK	24
3.1	Desk Study	24
3.2	Project Milestone	24
3.3	Tools and Equipment Used.....	26
3.4	Project Work	26
3.5	Sample Preparation & Testing.....	27
CHAPTER 4:	RESULTS AND DISCUSSION	28
4.1	Hardware	28
4.2	Software	29
4.3	Serial Interface	33
4.4	GSM Modem	35
4.5	Sending SMS	38
4.6	Peripheral Interface Controller (PIC) PIC16F877	42
4.7	Final Software- I-UTP Building Control v1.0	43
4.8	Visual Basic	48
4.9	Limitation of the System	51
CHAPTER 5:	CONCLUSION AND RECOMMENDATION	52
5.1	Conclusion	52
5.2	Recommendations	53
REFERENCES	54

APPENDICES

LIST OF FIGURES

- FIGURE 2.1** Events in the development of GSM
- FIGURE 2.2** GSM Network Architecture
- FIGURE 2.3** GSM Signaling Protocol Architecture
- FIGURE 2.4** Pin Layout of PIC 16F877
- FIGURE 2.5** Pin Configurations for Serial Port
- FIGURE 2.6** Overview of the System
- FIGURE 2.7** Nokia 3310
- FIGURE 2.8** Com Port Configuration
- FIGURE 2.9** Message Characteristics
- FIGURE 3.1** System Block Diagram
- FIGURE 4.1** Hardware Block Diagram
- FIGURE 4.2** MAX232 Configuration
- FIGURE 4.3** F-Bus/M-Bus Connection Pin in Nokia 3310
- FIGURE 4.4** Data Cable for Nokia 3310
- FIGURE 4.5** Data Cable Connection to Nokia 3310
- FIGURE 4.6** PIC Pin Configuration
- FIGURE 4.7** I-UTP Building Control Main Window
- FIGURE 4.8** Serial Connection Established Successfully
- FIGURE 4.9** F-Bus Connection Established Successfully
- FIGURE 4.10** Message Receive and Reply Sent
- FIGURE 4.11** The Terminal Logs

APPENDICES

- FIGURE A-1** Architecture of the PIC16F877 Microcontroller
- FIGURE A-2** Simplified Block Diagram of the PIC16F877 ADC Module
- FIGURE A-3** PIC16F877 Register File Map
- FIGURE A-4** Multisim Hardware Circuit
- FIGURE A-5** Hardware Test Board
- FIGURE A-6** Hardware Implementation on Vero Board
- FIGURE A-7** Checksum Program
- FIGURE A-8** Packer Program
- FIGURE A-9** Serial Interface Program
- FIGURE A-10** Main Page for i-UTP Building Control

- FIGURE A-11** Terminal Log for i-UTP Building Control
- FIGURE A-12** ASCII Code Window
- FIGURE A-13** Serial Information Window
- FIGURE A-14** How Data Is Transferred in MAX232 and Displayed In Oscilloscope
- FIGURE A-15** PIC C Language Program
- FIGURE A-16** Visual Basic Code
- FIGURE A-17** User Manual

ABBREVIATIONS AND NOMENCLATURES

1. GSM : Global System for Mobile Communication
2. SMS : Short Messaging Service
3. PIC : Peripheral Interface Controller
4. I/O : Input/Output
5. GPS : Global Positioning System
6. GIS : Geographical Information System

CHAPTER 1

INTRODUCTION

This chapter serves an overview of the communication system which emphasis on the communication systems available for Global System for Mobile Communication (GSM) transmission. A brief descriptions of the system used is discussed. The problem statement for the project is discussed on line-of-sight communication system. All the external factors that should be considered in designing a communication links are briefly explained. The specific objectives and scope of study of this project is discussed on the last section of this chapter.

1.1 BACKGROUND OF STUDY

Communication has always been an important element in our daily life. Nowadays, there are many method of conveying the information. Information can be delivered using medium such as wires and air. This project will go in depth into the world of wireless communication.

With the growing usage and rapid growth of the mobile phone technology and better coverage of the GSM services, a new area of study can be implemented which is controlling certain devices using Short Messaging Services (SMS). In order to accomplish this objective, knowledge about how the data is transferred in the GSM network is essential.

Knowledge on data protocol is also needed due to some mobile phone manufacturer who has created their own protocol in order to deliver the data in a GSM network. For this project, a specific manufacturer is chosen. Nokia is chosen because of its wide variety of mobile phone models and its availability in the market. The protocol that is being used by Nokia is called the F-Bus protocol.

1.2 PROBLEM STATEMENT

Mobile communication has developed and grows in Malaysia during the last three or four years. Nowadays, nearly every family has at least one mobile phone. The competition can also be seen between the phone manufacturer and also the service provider. Because of this competition, the services and coverage has also been upgraded and is more reliable. Many researches are done on how to improve and design new services using the existing technology. This project will use this expanding technology and try to incorporate it with another existing system in order to produce a better system.

Large buildings are not easy to maintain especially new buildings such as the new UTP buildings. They contain valuable machinery and confidential information. This building usually has its own control system. This control system is used to monitor the building area and its components.

This project will study and design an Intelligent Building Control and Monitoring System. This chosen building is the new UTP building. The system will use SMS technology and incorporate it with the building control system. The user can access the system by sending a command using their mobile phone SMS service. This is an easier way because the user does not have to check the building manually which will consume more time and energy. The system must also possess a security or safety features to avoid others from entering the system. Only authorized personnel will be able to use this system.

1.3 OBJECTIVES AND SCOPE OF STUDY

1.3.1 General Objectives of the Project

This Final Year Project course plays a vital role in achieving UTP's vision which is to produce a well rounded graduate. It is also a very great opportunity for students to relate the theoretical knowledge from class and applying it in project. Despite that, students will develop skills in work ethics, communication, management, interpersonal skills and etc. The objectives of the Final Year Project are:

- To develop a framework, this will enhance student's skills in the process of applying knowledge, expanding thoughts, solving problem independently and presenting findings.
- Develop a system that can locate a vehicle using communication system preferably wireless communication.
- To produce a system that is reliable and can be easily handled by other people and also low in cost if possible.
- To integrate the hardware and the software part of the system to make it easier to handled and managed.

1.3.2 Specific Objectives of the Project

The specific objectives of the project are:

- To design a building control system using Short Messaging Service(SMS) through the usage of Nokia F-Bus protocol
- To acquire knowledge on Programmable Integrated Circuit (PIC) programming using MPLAB and its hardware implementation.
- To enhance knowledge on digital circuit and real-time data communication in real-world application.

1.3.2 Scope of Study

The scope of study for this project is the potential of integrating GSM network and its services with other existing system. In order to fully utilize the GSM network and its services, further knowledge about the network and how it work using its own protocol must be gained. Other than that, more advance knowledge about programming language must be acquired in order to program the controller using C language and designing the software using Visual Basic. Some circuit designing skill must also be acquired in order to build a model to represent the whole system at a smaller scale.

CHAPTER 2

LITERATURE REVIEW/THEORY

This chapter discussed on the theories and literature review of the project. Among the theories that will be discussed are the history of GSM, what is PIC and its advantages, how the serial interface is used, the Nokia F-Bus characteristics and the overview of the proposed system.

2.1 Global System for Mobile Communication (GSM)

2.1.1 History of GSM

During the early 1980s, analog cellular telephone systems were experiencing rapid growth in Europe, particularly in Scandinavia and the United Kingdom, but also in France and Germany. Each country developed its own system, which was incompatible with everyone else's in equipment and operation. This was an undesirable situation, because not only was the mobile equipment limited to operation within national boundaries, which in a unified Europe were increasingly unimportant, but there was a very limited market for each type of equipment, so economies of scale, and the subsequent savings, could not be realized.

The Europeans realized this early on, and in 1982 the Conference of European Posts and Telegraphs (CEPT) formed a study group called the Groupe Spécial Mobile (GSM) to study and develop a pan European public land mobile system.

The proposed system had to meet certain criteria:

- good subjective speech quality,
- low terminal and service cost,
- support for international roaming,
- ability to support handheld terminals,
- support for range of new services and facilities,
- spectral efficiency, and
- ISDN compatibility.

In 1989, GSM responsibility was transferred to the European Telecommunication Standards Institute (ETSI), and phase I of the GSM specifications were published in 1990. Commercial service was started in mid1991, and by 1993 there were 36 GSM networks in 22 countries, with 25 additional countries having already selected or considering GSM. This is not only a European standard - South Africa, Australia, and many Middle and Far East countries have chosen GSM. By the beginning of 1994, there were 1.3 million subscribers worldwide. The acronym GSM now (aptly) stands for Global System for Mobile telecommunications.

The developers of GSM chose an unproven (at the time) digital system, as opposed to the then as standard analog cellular systems like AMPS in the United States and TACS in the United Kingdom. They had faith that advancements in compression algorithms and digital signal processors would allow the fulfillment of the original criteria and the continual improvement of the system in terms of quality and cost. The 8000 pages of the GSM recommendations try to allow flexibility and competitive innovation among suppliers, but provide enough guidelines to guarantee the proper interworking between the components of the system. This is done in part by providing descriptions of the interfaces and functions of each of the functional entities defined in the system.

2.1.2 Services Provided by GSM

From the beginning, the planners of GSM wanted ISDN compatibility in services offered and control signaling used. The radio link imposed some limitations, however, since the standard ISDN bit rate of 64 kbps could not be practically achieved.

Using the ITUT definitions, telecommunication services can be divided into bearer services, teleservices, and supplementary services. The digital nature of GSM allows data, both synchronous and asynchronous, to be transported as a bearer service to or from an ISDN terminal. Data can use either the transparent service, which has a fixed delay but no guarantee of data integrity, or a nontransparent service, which guarantees data integrity through an Automatic Repeat Request (ARQ) mechanism, but with a variable delay. The data rates supported by GSM are 300 bps, 600 bps, 1200 bps, 2400 bps, and 9600 bps.

The most basic teleservices supported by GSM is telephony. There is an emergency service, where the nearest emergency service provider is notified by dialing three digits (similar to 911). Group 3 fax, an analog method described in ITUT recommendation T.30, is also supported by use of an appropriate fax adaptor. A unique feature of GSM compared to older analog systems is the Short Message Service (SMS). SMS is a bidirectional service for sending short alphanumeric (up to 160 bytes) messages in a store and forward fashion. For point to point SMS, a message can be sent to another subscriber to the service, and an acknowledgement of receipt is provided to the sender. SMS can also be used in a cell broadcast mode, for sending messages such as traffic updates or news updates. Messages can be stored in the SIM card for later retrieval.

Supplementary services are provided on top of teleservices or bearer services, and include features such as caller identification, call forwarding, call waiting, multiparty conversations, and barring of outgoing (international) calls, among others.

Year	Events
1982	CEPT establishes a GSM group in order to develop the standards for a pan-European cellular mobile system
1985	Adoption of a list of recommendations to be generated by the group
1986	Field tests were performed in order to test the different radio techniques proposed for the air interface
1987	TDMA is chosen as access method (in fact, it will be used with FDMA) Initial Memorandum of Understanding (MoU) signed by telecommunication operators (representing 12 countries)
1988	Validation of the GSM system
1989	The responsibility of the GSM specifications is passed to the ETSI
1990	Appearance of the phase 1 of the GSM specifications
1991	Commercial launch of the GSM service
1992	Enlargement of the countries that signed the GSM- MoU Coverage of larger cities/airports
1993	Coverage of main roads GSM services start outside Europe
1995	Phase 2 of the GSM specifications Coverage of rural areas

Figure 2.1: Events in the development of GSM

2.1.3 Architecture of the GSM network

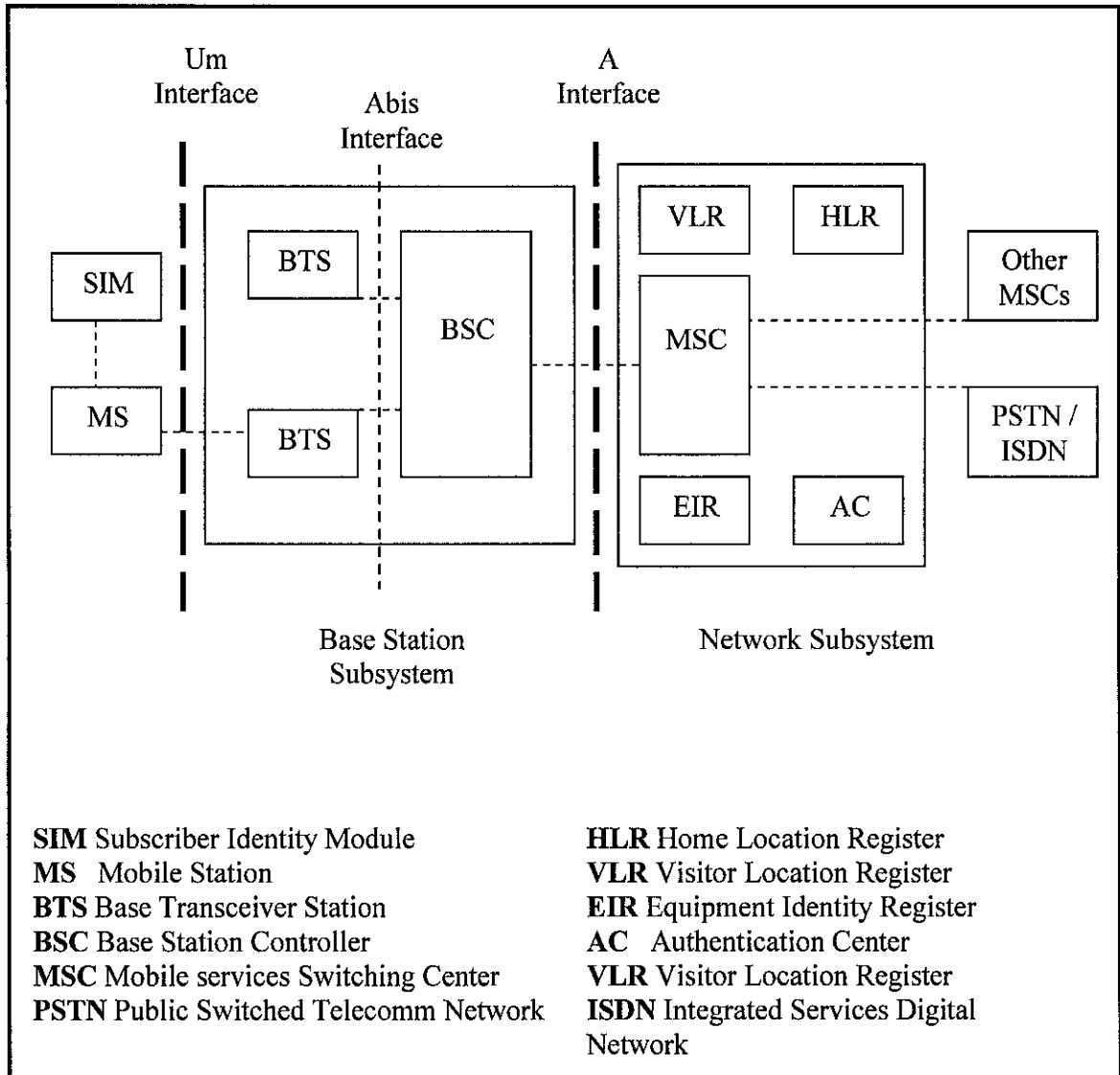


Figure 2.2 GSM Network Architecture

A GSM network is composed of several functional entities, whose functions and interfaces are defined. Figure 2.2 shows the layout of a generic GSM network. The GSM network can be divided into three broad parts. The Mobile Station is carried by the subscriber; the Base Station Subsystem controls the radio link with the Mobile Station. The Network Subsystem, the main part of which is the Mobile services Switching Center, performs the switching of calls between the mobile and other fixed or mobile network users, as well as management of mobile services, such as authentication. Not shown is the Operations and Maintenance center, which oversees the proper

operation and setup of the network. The Mobile Station and the Base Station Subsystem communicate across the Um interface, also known as the air interface or radio link. The Base Station Subsystem communicates with the Mobile service Switching Center across the A interface.

2.1.3.1 Mobile Station

The mobile station (MS) consists of the physical equipment, such as the radio transceiver, display and digital signal processors, and a smart card called the Subscriber Identity Module (SIM). The SIM provides personal mobility, so that the user can have access to all subscribed services irrespective of both the location of the terminal and the use of a specific terminal. By inserting the SIM card into another GSM cellular phone, the user is able to receive calls at that phone, make calls from that phone, or receive other subscribed services.

The mobile equipment is uniquely identified by the International Mobile Equipment Identity (IMEI). The SIM card contains the International Mobile Subscriber Identity (IMSI), identifying the subscriber, a secret key for authentication, and other user information. The IMEI and the IMSI are independent, thereby providing personal mobility. The SIM card may be protected against unauthorized use by a password or personal identity number.

2.1.3.2 Base Station Subsystem

The Base Station Subsystem is composed of two parts, the Base Transceiver Station (BTS) and the Base Station Controller (BSC). These communicate across the specified Abis interface, allowing (as in the rest of the system) operation between components made by different suppliers.

The Base Transceiver Station houses the radio transceivers that define a cell and handles the radio link protocols with the Mobile Station. In a large urban area, there will potentially be a large number of BTSs deployed. The requirements for a BTS are ruggedness, reliability, portability, and minimum cost.

The Base Station Controller manages the radio resources for one or more BTSs. It handles radio channel setup, frequency hopping, and handovers, as described below. The BSC is the connection between the mobile and the Mobile service Switching Center (MSC). The BSC also translates the 13 kbps voice channel used over the radio link to the standard 64 kbps channel used by the Public Switched Telephone Network or ISDN.

2.1.3.3 Network Subsystem

The central component of the Network Subsystem is the Mobile services Switching Center (MSC). It acts like a normal switching node of the PSTN or ISDN, and in addition provides all the functionality needed to handle a mobile subscriber, such as registration, authentication, location updating, handovers, and call routing to a roaming subscriber. These services are provided in conjunction with several functional entities, which together form the Network Subsystem. The MSC provides the connection to the public fixed network (PSTN or ISDN), and signaling between functional entities uses the ITUT Signaling System Number 7 (SS7), used in ISDN and widely used in current public networks.

The Home Location Register (HLR) and Visitor Location Register (VLR), together with the MSC, provide the call routing and (possibly international) roaming capabilities of GSM. The HLR contains all the administrative information of each subscriber registered in the corresponding GSM network, along with the current location of the mobile. The current location of the mobile is in the form of a Mobile

Station Roaming Number (MSRN) which is a regular ISDN number used to route a call to the MSC where the mobile is currently located. There is logically one HLR per GSM network, although it may be implemented as a distributed database.

The Visitor Location Register contains selected administrative information from the HLR, necessary for call control and provision of the subscribed services, for each mobile currently located in the geographical area controlled by the VLR. Although each functional entity can be implemented as an independent unit, most manufacturers of switching equipment implement one VLR together with one MSC, so that the geographical area controlled by the MSC corresponds to that controlled by the VLR, simplifying the signaling required. Note that the MSC contains no information about particular mobile stations - this information is stored in the location registers.

The other two registers are used for authentication and security purposes. The Equipment Identity Register (EIR) is a database that contains a list of all valid mobile equipment on the network, where each mobile station is identified by its International Mobile Equipment Identity (IMEI). An IMEI is marked as invalid if it has been reported stolen or is not type approved. The Authentication Center is a protected database that stores a copy of the secret key stored in each subscriber's SIM card, which is used for authentication and ciphering of the radio channel.

2.1.4 Radio link aspects

The International Telecommunication Union (ITU), which manages the international allocation of radio spectrum (among other functions) allocated the bands 890-915 MHz for the uplink (mobile station to base station) and 935-960 MHz for the downlink (base station to mobile station) for mobile networks in Europe. Since this range was already being used in the early 1980s by the analog systems of the day, the CEPT had the foresight to reserve the top 10 MHz of each band for the GSM network that was still being developed. Eventually, GSM will be allocated the entire 2x25 MHz bandwidth.

Since radio spectrum is a limited resource shared by all users, a method must be devised to divide up the bandwidth among as many users as possible. The method chosen by GSM is a combination of Time and Frequency Division Multiple Access (TDMA/FDMA). The FDMA part involves the division by frequency of the total 25 MHz bandwidth into 124 carrier frequencies of 200 kHz bandwidth. One or more carrier frequencies are then assigned to each base station. Each of these carrier frequencies is then divided in time, using a TDMA scheme, into eight time slots. One time slot is used for transmission by the mobile and one for reception. They are separated in time so that the mobile unit does not receive and transmit at the same time, a fact that simplifies the electronics.

2.1.5 Network aspects

Ensuring the transmission of voice or data of a given quality over the radio link is only half the problem in a cellular mobile network. The fact that the geographical area covered by the network is divided into cells necessitates the implementation of a handover mechanism. Also, the fact that the mobile can roam nationally and internationally in GSM requires that registration, authentication, call routing and location updating functions exist in the GSM network.

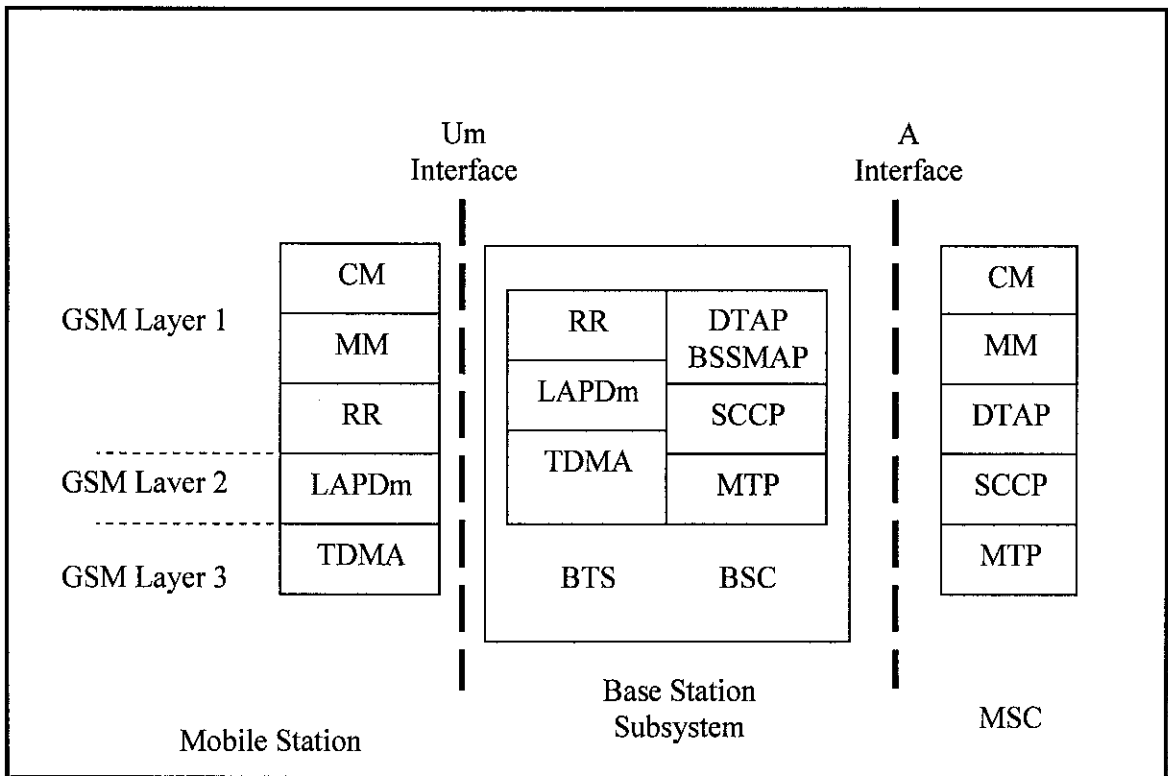


Figure 2.3: GSM Signaling Protocol Architecture

The signaling protocol in GSM is structured in three layers shown in Figure 2.3. Layer 1 is the physical layer, which uses the channel structures discussed above. Layer 2 is the data link layer. Across the Um interface, the data link layer uses a slight modification of the LAPD protocol used in ISDN, called LAPDm. Across the A interface, the lower parts of Signaling System Number 7 are used. Layer 3 is subdivided into 3 sub layers.

Radio Resources Management

Controls the setup, maintenance, and termination of radio channels

Mobility Management

Manages the location updating, handovers, and registration procedures, discussed below

Connection Management

Handles general call control, similar to CCITT Recommendation Q.931, and provides supplementary services.

Signaling between the different entities in the network, such as between the HLR and VLR, is accomplished through the Mobile Application Part (MAP). Application parts are the top layer of Signaling System Number 7. The specification of the MAP is complex. It is one of the longest documents in the GSM recommendations, said to be over 600 pages in length.

2.2 Peripheral Interface Controller (PIC)

The PIC is a high performance RISC CPU. It operates at 4MHz and 25ms instruction per cycle. It contains three type of memory which is the FLASH Program Memory, Data Memory (RAM) and EEPROM Data Memory. The PIC16F877 is a high-performance FLASH microcontroller that provides engineers with the highest design flexibility possible. In addition to 8192x14 words of FLASH program memory, 256 data memory bytes, and 368 bytes of user RAM, PIC16F877 also features an integrated 8-channel 10-bit Analogue-to-Digital converter. Peripherals include two 8-bit timers, one 16-bit timer, a Watchdog timer, Brown-Out-Reset (BOR), In-Circuit-Serial Programming™, RS-485 type UART for multi-drop data acquisition applications, and I2C™ or SPI™ communications capability for peripheral expansion. Precision timing interfaces are accommodated through two CCP modules and two PWM modules.

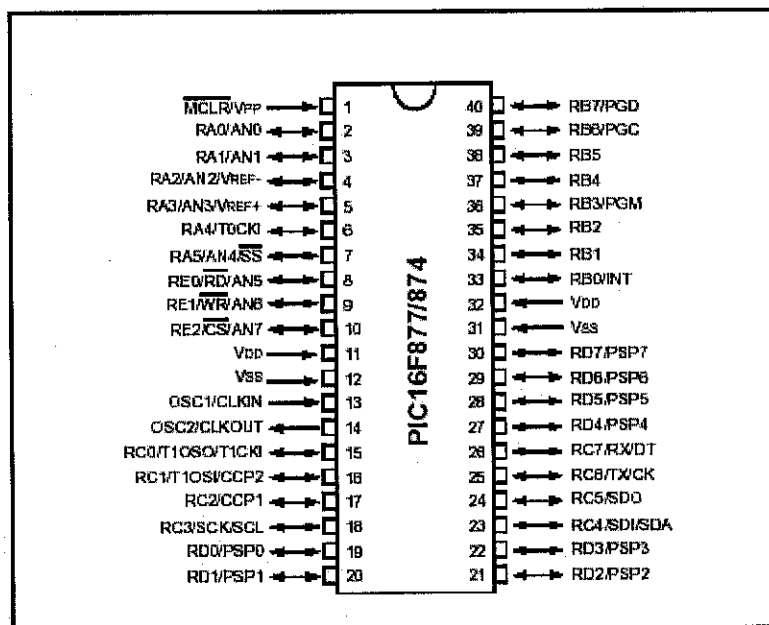


Figure 2.4: Pin Layout of PIC 16F877

In this project, the PIC will be used as a controller. It will control the building's lamps, air-conditioner and doors. The base station or terminal will communicate with the PIC in order to know the status of each device and to control them. The overall architecture of the PIC can be seen in **Appendix A-1**.

2.3 Serial interface

The serial port is an I/O device. An I/O device is just a way to get data into and out of a computer. There are many types of I/O devices such as serial ports, parallel ports, disk drive controllers, Ethernet boards, universal serial buses and many others. Most PC's have one or two serial ports. Each has a 9-pin connector or sometimes 25-pin on the back of the computer. Computer programs can send data (bytes) to the transmit pin (output) and receive bytes from the receive pin (input). The other pins are for control purposes and ground.

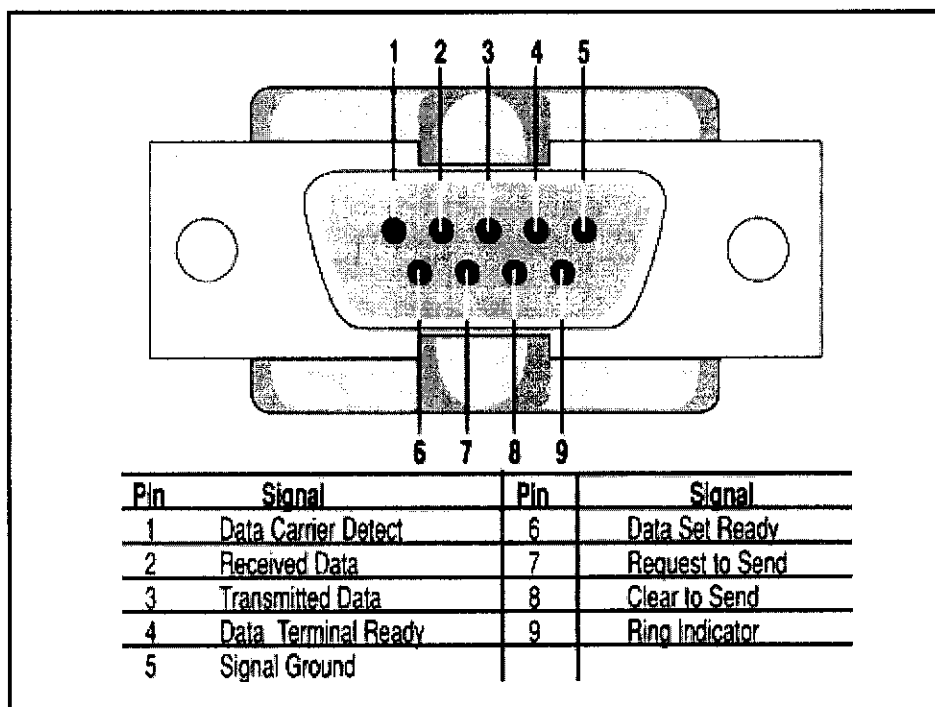


Figure 2.5: Pin Configuration for Serial Port

The serial port is much more than just a connector. It converts the data from parallel to serial and changes the electrical representation of the data. Inside the computer, data bits flow in parallel, using many wires at the same time. Serial flow is a stream of bits over a single wire, such as on the Transmit or Receive pin of the serial connector. For the serial port to create such a flow, it must convert data from parallel inside the computer to serial on the transmit pin and conversely.

The serial port is harder to interface than the parallel port. In most cases, any device connected to the serial port will need the serial transmission converted back to parallel so that it can be used. This can be done using a UART. On the software side of things, there are many more registers that have to be attended to than on a standard parallel port.

The advantages of using serial data transfer rather than parallel are:

- Less wires than parallel transmission. If your device needs to be mounted a far distance away from the computer then 3 core cable (Null Modem Configuration) is going to be a lot cheaper than running 19 or 25 core cable. However you must take into account the cost of the interfacing at each end.
- Microcontrollers have also proven to be quite popular recently. Many of these have in built SCI (Serial Communications Interfaces) which can be used to talk to the outside world. Serial communication reduces the pin count of these MPU's. Only two pins are commonly used, Transmit Data (TXD) and Receive Data (RXD) compared with at least 8 pins using an 8 bit Parallel method. Furthermore, it may also require a Strobe.
- Serial cables can be longer than parallel cables. The serial port transmits a '1' as -3 to -25 volts and a '0' as +3 to +25 volts where as a parallel port transmits a '0' as 0v and a '1' as 5v. Therefore the serial port can have a maximum swing of 50V compared to the parallel port which has a maximum swing of 5 Volts. Therefore cable loss is not going to be as much of a problem for serial cables as they are for parallel.

2.4 Overview of the System

Figure 2.6 shows the overview of the proposed system. To use the system user must have a mobile phone and the system uses short messaging services (SMS) to send commands to the terminal located at each building. Every building will have its own terminal and different ID that will enable the user to control each building separately. This is done for the security reason. Unauthorized personnel can't access the system. The basic idea is to use the GSM network to convey the message from the user to the controller.

There will be a main controller that will control the devices while a special software to convert the message from the user into instructions that the controller understands. A mobile phone will be connected to the computer or terminal. This hand phone serves as a GSM modem that will receive messages from users and sends the reply to the user. The terminal is equipped with special software called i-UTP Building Control that will convert the message into instructions that the I/O controller understands. I/O controller is made up from the PIC16F877. The I/O controller is connected to the devices such as air-conditioner, doors and lamps. The terminal will process the SMS message or command sent by the user and gives instruction to the I/O controller to do the necessary action requested by the user. The detail operation of the system will be discussed in the discussion section.

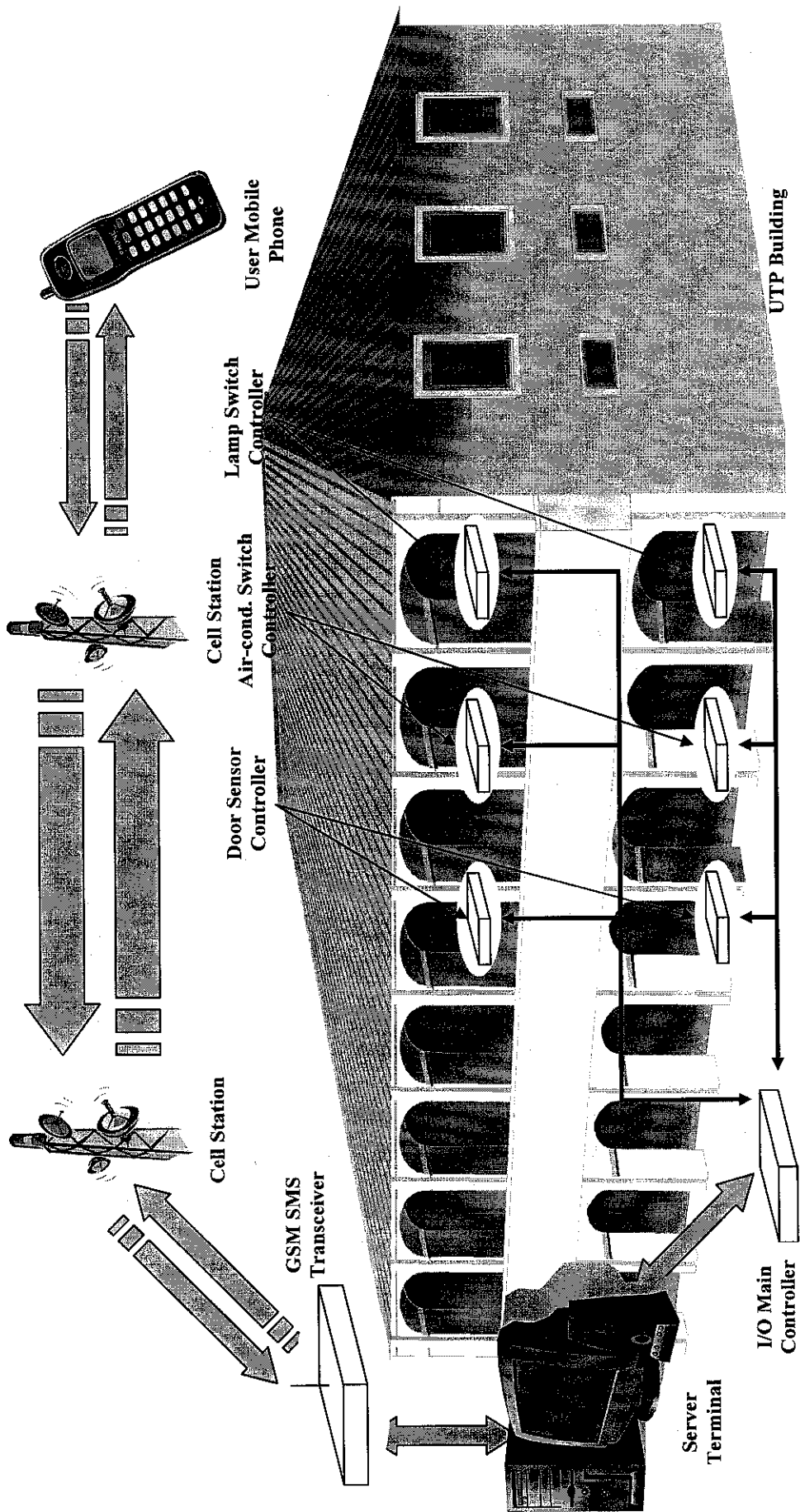


Figure 2.6: Overview of the Project

2.5 Nokia 3310 and F-Bus Protocol

For this project, Nokia 3310 is used because of its availability, cost and functions. Furthermore the data cable can be easily purchased. The F-Bus protocol is owned by Nokia and only Nokia phones use this protocol. The protocol allows the user to explore the phone capability and use it to interface with other software. This sms feature is utilized in the project mainly sending and receiving sms.



Figure 2.7: Nokia 3310

The mobile phone will act as a gsm modem that will send and receive the messages from the user and controller. The message cost depends on the service provider rate. The mobile phone is connected to the terminal or computer via a data cable that can be connected through the serial port of the computer. In order to make this project successful, a minimum number of two mobile phones are needed. One serves as the GSM modem and the other is used by the user to give the appropriate commands. The detailed explanation on how the phone actually works in this project will be discussed in the discussion chapter.

2.6 Nokia F-Bus Protocols Characteristics

The F-Bus is designed by Nokia for the phone to interact with a computer. It has its own protocol. To setting to properly configure the com port are as follows:

speed	115 200
num bits	8
parity	none
Stop bit	1

Figure 2.8: Com Port Configuration

A standard message looks like this one:

```
1E 02 00 04 00 0B 01080002010463020401 40 00 3900
```

This corresponds to:

```
[Frame type(1)][Src dev(1)][Dst dev(1)][CMD(1)][Frame  
type(1)][Len(1)][DATA(X)][Seq(1)][Padd (1 or  
0)][Chksum(2)]
```

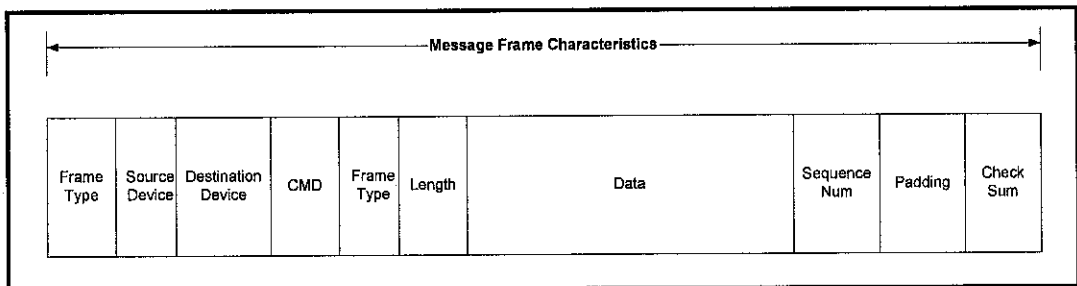


Figure 2.9: Message Characteristics

2.5.1 Frame type

The frame type indicates which type of protocol is using:

- 1E : Serial F-bus frame
- 1c : Irda F-bus frame

2.5.2 Source device and Destination device

Indicate the source and the destination device

- 02 Phone
- 00 Computer

2.5.3 CMD

This is the command type, it define which type of information is about.

- need a fix

2.5.4 Frame type

Used if the message exceeded 255 then it give which part is sending.

2.5.5 Length

The length of the packet. To calculate it: Data + Sequence number. So in other word: $\text{length} = \text{data} + 1$ (in hex)

2.5.6 Data

The packed data.

2.5.7 Sequence number for regular packet

The sequence number for the standard frame seems to be between 40 up to 47. So always initialize it to 40 at the beginning seems to be working.

2.5.8 Padding

Since the packet as to be an odd number, if the length is even it as to be added. The padding is always 00.

2.5.9 Checksum

The check sum is in fact two different checksum. The first hex represents the XOR of all the odd hex block from the packet, the second represent the XOR of all the even Hex block of the packet.

CHAPTER 3

METHODOLOGY/PROJECT WORK

The methodology on how the project was conducted is discussed in this chapter. The preliminary research was conducted to get the overview of the topic and to design the milestone or Gantt chart. This project was planned to be completed in two semester, where the first half of the semester was to concentrate on how the data will be sent from one place to another and the basic structure of the system. The real circuit and implementation of the system was designed in the second half of the semester. The project work uses PIC and mainly Visual Basic to develop the software.

3.1 DESK STUDY

Desk study plays significant impact to strengthen the basic knowledge about anything related to the project. Internet is the main source for the study, as well as referring to books, journals, articles and reports. Visual Basic and C language need to be self-studied in order to design the software for the project. Other than programming, knowledge about designing and constructing circuit must also be studied.

3.2 PROJECT MILESTONE

The student as well as the supervisor can easily monitor the progress of the project. Since this project is for two semester project, the milestone should be planned in such a way that the time is enough to complete the overall task planned for the two semesters.

The overall suggested milestone is in Appendices **Figure B-1** and **Figure B-2**. A summary of the project phases can be listed as follows;

- Phase 1: Planning Phase
- Phase 2: Research and Literature Review
- Phase 3: Designing Theoretical Circuit
- Phase 4: Implementing Practical Circuit
- Phase 5: Final Testing and Documentation

Phase 1 of the project involves the planning of a specific outline of the proposed work, requirements, and goals of the project. A Gantt chart is produced as a guide for the student as well as the supervisor to complete this project.

Phase 2 encompassed a literature review and background research on the topic, the determination of resources requirements, the division of the project into logical steps, and the choosing of a methodology and implementation process for the completion of the project.

Phase 3 is the design process and testing process. This includes designing of the theoretical circuit and testing the circuit. Some PIC programming will also be included as well as designing the software using Visual Basic.

Phase 4 will involve implementing the theoretical circuit on to the test board and debugging the circuit. Only after this process the circuit can be transferred into the real board.

The final phase of the project includes final testing on the software and documentation of the project. At this phase, the project is expected to be completed and in working condition.

3.3 TOOLS AND EQUIPMENT USED

Visual Basic is the main software that is used in this project to develop the software while the PIC uses the WARP13 software to program it. The compiler used for PIC is PIC C compiler. Other than these software, basic components are used to construct the hardware.

3.4 PROJECT WORK

The main focus is to develop the software using Visual Basic which will integrate the sms from telephone to the terminal and the connection from terminal to the PIC. In order to achieve this objective, the program was divided into smaller functional program (**Appendices A-4 and Appendix A-5**).

Beside from the software programming, the PIC also needs to be program so that it can be used with the test board that has been constructed. After the board is tested, the circuit has to be transferred to the real board. The final stage is testing the whole system and makes correction if there are errors.

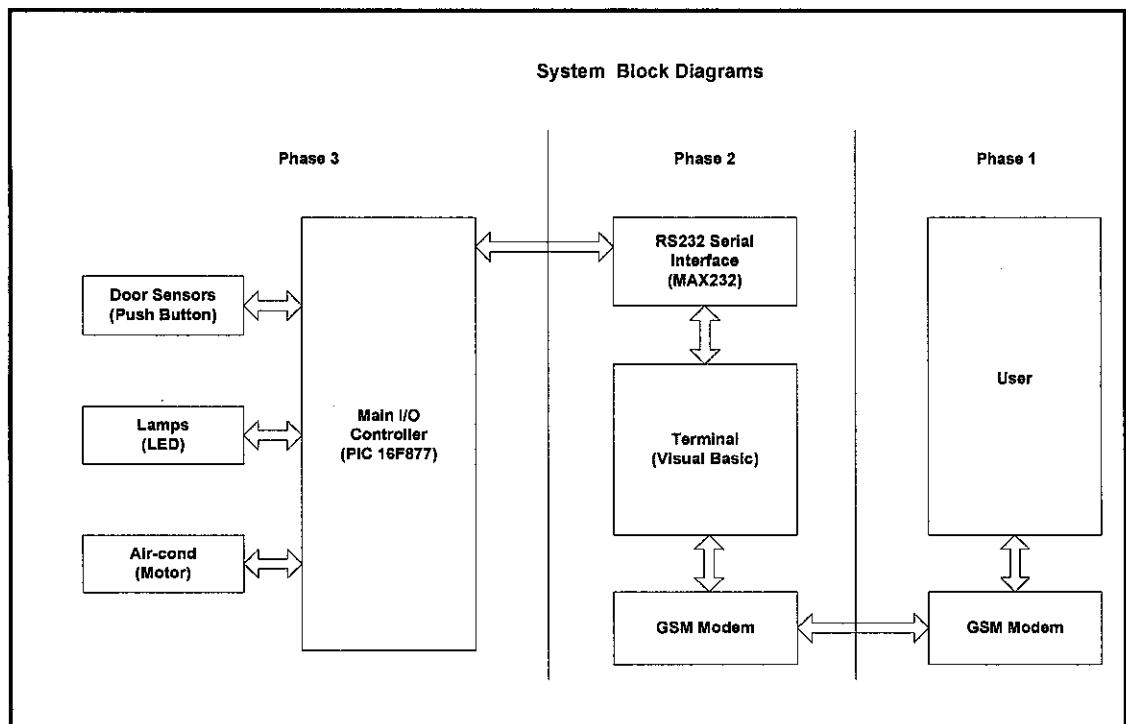


Figure 3.1: System Block Diagram

3.5 SAMPLE PREPARATION & TESTING

There are three different types of input samples being used in the process of developing the simulation program. Each sample is tested to verify that the program algorithm work and give the expected output. This bottom-up testing is important to make sure that each subprograms work correctly integrates them.

At the first stage, the input samples are defined manually with a limited number of input sequences. At this stage, the purpose of the input samples is to verify the functionality (black-box testing) either the subprograms will give the correct expected outputs. The input of the subsystems also might be from the other subsystem outputs. In this case, the lower level programs are tested first before move to the higher level programs.

The testing process occurred at each stage of the project. After a small part of the project is finish, then some testing is done to make sure that the part is working. The final testing is done when the final product is completed. The testing covers different command sent or received other than the specified commands and using different models and brand of hand phone.

CHAPTER 4

RESULTS AND DISCUSSION

This chapter discussed on detail theory behind the construction of the hardware and software of the project. Other than that, the final product is also included and discussed. The steps on how to use the final product is also included in this chapter. This includes the SMS commands that will monitor and control the dedicated devices in the building.

4.1 Hardware

The test circuit have been constructed and tested with a test program. The real testing can be done when the software is fully developed. The layout of the circuit can be seen in **Appendix A-4**.

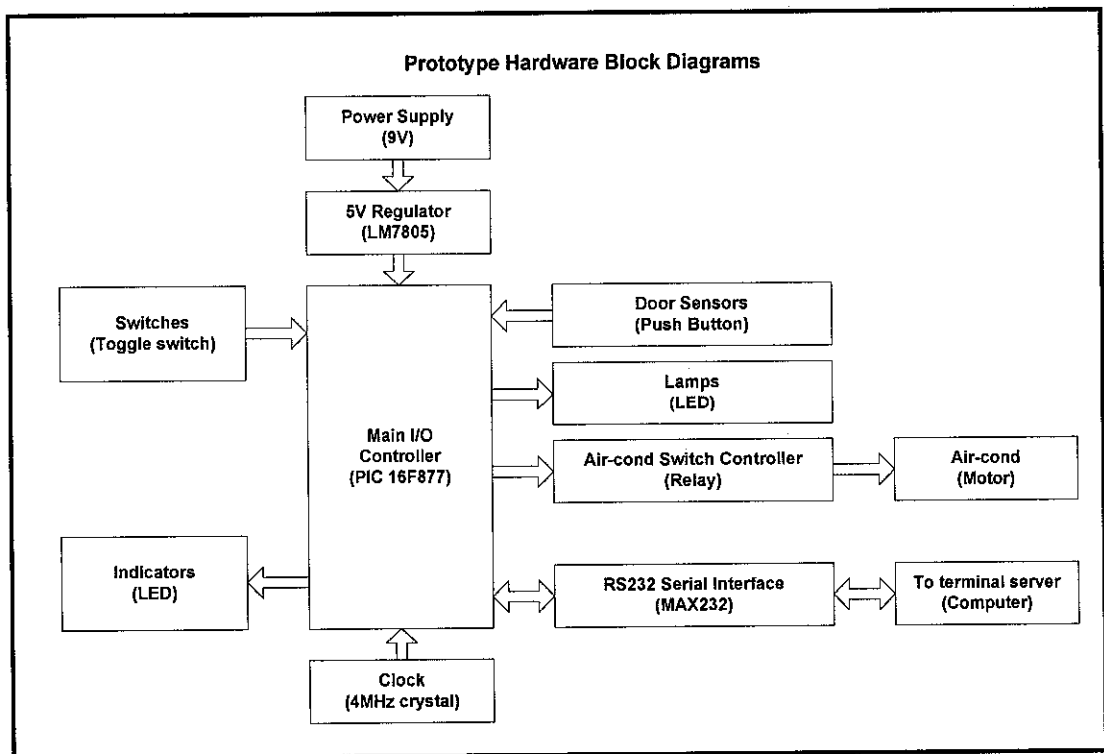


Figure 4.1: Hardware Block Diagrams

The Figure 4.1 describes the configuration of the hardware; how each device is connected to each other and what they represent in the real system. Among the components used to construct the hardware are:

- Motor
- Relay
- MAX232
- PIC16F877
- 4Mhz Crystal Oscillator
- LEDs
- Switch
- LM7805 Voltage Regulator
- Push Button

The circuit is first constructed on the bread board. This is done to see and correct the error before implementing the circuit into the final product. The test board can be viewed in **Appendix A-5**. When the circuit has been tested and is operational, then it was transferred into the vero board.

4.2 Software

The development of the software requires a lot of phases. Before the real product can be produced, there are many test and small program that have to be created and tested. All of these small programs will then be group and combined to form the final software. The test program is done in C language but the final program will be in Visual Basic.

4.2.1 Subprograms

Several subprograms have been created. These subprograms will be combined in the final program. Each subprogram has its own functions. The codes can be found in **Appendix A-7** and **Appendix A-8**.

The packer program is used to pack the 7 bit data into an 8 bit data. This is crucial because the F-Bus system only used 8 bit configuration. If a 7 bit data is used, then the decoded data will not be the same with the data that have been sent.

Meanwhile the Check Sum program is used to calculate the parity of the odd and even bits in each message frame. This is a kind of safety measure because the message frame is sent with the check sum and the receiver must calculate the parity again and verify the result with the sent checksum. By doing this, any loss of data can be detected. The print screen of the final software can be seen in **Appendix A-10** to **Appendix A-13**.

4.2.2 Commands Sets

These commands sets are created to help design the software. This syntax will be used in the system and only these commands will be recognize by the SMS terminal.

4.2.2.1 User->Terminal Command Set

System Check

A1. I-UTP ACAD22 System Info

Status Check

- B1. I-UTP ACAD22-01-04 Status Lamp 1
- B2. I-UTP ACAD22-01-04 Status Lamp 2
- B3. I-UTP ACAD22-01-04 Status Aircond 1
- B4. I-UTP ACAD22-01-04 Status Aircond 2
- B5. I-UTP ACAD22-01-04 Status Door 1
- B6. I-UTP ACAD22-01-04 Status Door 2
- B7. I-UTP ACAD22-01-04 Status Room-Door 1
- B8. I-UTP ACAD22-01-04 Status Room-Door 2

Switch Control

- C1. I-UTP ACAD22-01-04 Switch Lamp 1 ON
- C2. I-UTP ACAD22-01-04 Switch Lamp 1 OFF
- C3. I-UTP ACAD22-01-04 Switch Lamp 2 ON
- C4. I-UTP ACAD22-01-04 Switch Lamp 2 OFF
- C5. I-UTP ACAD22-01-04 Switch Aircond 1 ON
- C6. I-UTP ACAD22-01-04 Switch Aircond 1 OFF
- C7. I-UTP ACAD22-01-04 Switch Aircond 2 ON
- C8. I-UTP ACAD22-01-04 Switch Aircond 2 OFF

4.2.2.2 Terminal->User Command Set

System Check

A1. I-UTP ACAD22 System OK

Status Check

- B1. I-UTP ACAD22-01-04 Lamp 1 ON
- B1. I-UTP ACAD22-01-04 Lamp 1 OFF
- B2. I-UTP ACAD22-01-04 Lamp 2 ON
- B2. I-UTP ACAD22-01-04 Lamp 2 OFF
- B3. I-UTP ACAD22-01-04 Aircond 1 ON
- B3. I-UTP ACAD22-01-04 Aircond 1 OFF
- B4. I-UTP ACAD22-01-04 Aircond 2 ON
- B4. I-UTP ACAD22-01-04 Aircond 2 OFF
- B5. I-UTP ACAD22-01-04 Door 1 Opened
- B5. I-UTP ACAD22-01-04 Door 1 Closed
- B6. I-UTP ACAD22-01-04 Door 2 Opened
- B6. I-UTP ACAD22-01-04 Door 2 Closed
- B7. I-UTP ACAD22-01-04 Room Door 1 Opened
- B7. I-UTP ACAD22-01-04 Room Door 1 Closed
- B8. I-UTP ACAD22-01-04 Room Door 2 Opened
- B8. I-UTP ACAD22-01-04 Room Door 2 Closed

Switch Control

- C1. I-UTP ACAD22-01-04 Lamp 1 has been switched ON
- C1. I-UTP ACAD22-01-04 Lamp 1 has been switched OFF
- C1. I-UTP ACAD22-01-04 Request denied. Lamp 1 is already ON
- C1. I-UTP ACAD22-01-04 Request denied. Lamp 1 is already OFF
- C2. I-UTP ACAD22-01-04 Lamp 2 has been switched ON
- C2. I-UTP ACAD22-01-04 Lamp 2 has been switched OFF
- C2. I-UTP ACAD22-01-04 Request denied. Lamp 2 is already ON
- C2. I-UTP ACAD22-01-04 Request denied. Lamp 2 is already OFF
- C3. I-UTP ACAD22-01-04 Aircond 1 has been switched ON
- C3. I-UTP ACAD22-01-04 Aircond 1 has been switched OFF
- C3. I-UTP ACAD22-01-04 Request denied. Aircond 1 is already ON
- C3. I-UTP ACAD22-01-04 Request denied. Aircond 1 is already OFF

- C4. I-UTP ACAD22-01-04 Aircond 2 has been switched ON
- C4. I-UTP ACAD22-01-04 Aircond 2 has been switched OFF
- C4. I-UTP ACAD22-01-04 Request denied. Aircond 2 is already ON
- C4. I-UTP ACAD22-01-04 Request denied. Aircond 2 is already OFF

4.2.2.3 Terminal->Controller Command Set

Switch Control

	Start	cmdgrp	block	floor	room	device	dev no	on/off	stop
C1.	0xAA	0x03	0x16	0x01	0x04	0x00	0x00	0x01	0xFF
C2.	0xAA	0x03	0x16	0x01	0x04	0x00	0x00	0x00	0xFF
C3.	0xAA	0x03	0x16	0x01	0x04	0x00	0x01	0x01	0xFF
C4.	0xAA	0x03	0x16	0x01	0x04	0x00	0x01	0x00	0xFF
C5.	0xAA	0x03	0x16	0x01	0x04	0x01	0x00	0x01	0xFF
C6.	0xAA	0x03	0x16	0x01	0x04	0x01	0x00	0x00	0xFF
C7.	0xAA	0x03	0x16	0x01	0x04	0x01	0x01	0x01	0xFF
C8.	0xAA	0x03	0x16	0x01	0x04	0x01	0x01	0x00	0xFF

4.2.2.4 Controller->Terminal Command Set

Switch Control

If execution succeeds:

	Device	dev no	on/off
C1.	0x00	0x00	0x01
C2.	0x00	0x00	0x00
C3.	0x00	0x01	0x01
C4.	0x00	0x01	0x00
C5.	0x01	0x00	0x01
C6.	0x01	0x00	0x00
C7.	0x01	0x01	0x01
C8.	0x01	0x01	0x00

4.2.2.5 Error Reply

If I-UTP is not sent as a prefix
(No reply)

If wrong command is sent
I-UTP ACAD22-01-04 Unidentified request

4.3 Serial Interface

In order for the PIC to communicate with the terminal, a serial interface must be developed using Visual Basic. A tester program has been developed for testing purposes. The screen shot of the interface is shown in **Appendix A-9**. This tester program is used to test the serial communication between PIC and the terminal. The program was developed using Visual Basic

The serial port is an Asynchronous port which transmits one bit of data at a time, usually connecting to the UART Chip. Serial Ports are commonly found on the majority of PC Compatible computers In order to achieve this communication, a special IC is uses called MAX232 from Maxim or RS232 from RS. RS232 signals are represented by voltage levels with respect to a system common (power / logic ground). The "idle" state (MARK) has the signal level negative with respect to common, and the "active" state (SPACE) has the signal level positive with respect to common. RS232 has numerous handshaking lines (primarily used with modems), and also specifies a communications protocol.

The RS-232 interface presupposes a common ground between the DTE and DCE. This is a reasonable assumption when a short cable connects the DTE to the DCE, but with longer lines and connections between devices that may be on different electrical busses with different grounds, this may not be true.

RS232 data is bi-polar; +3 TO +12 volts indicate an "ON or 0-state (SPACE) condition" while A -3 to -12 volts indicates an "OFF" 1-state (MARK) condition.

Modern computer equipment ignores the negative level and accepts a zero voltage level as the "OFF" state. In fact, the "ON" state may be achieved with lesser positive potential. This means circuits powered by 5 VDC are capable of driving RS232 circuits directly; however, the overall range that the RS232 signal may be transmitted / received may be dramatically reduced.

The output signal level usually swings between +12V and -12V. The "dead area" between +3v and -3v is designed to absorb line noise. In the various RS-232-like definitions this dead area may vary. For instance, the definition for V.10 has a dead area from +0.3v to -0.3v. Many receivers designed for RS-232 are sensitive to differentials of 1v or less.

Data is transmitted and received on pins 2 and 3 respectively. Data Set Ready (DSR) is an indication from the Data Set (i.e., the modem or DSU/CSU) that it is on. Similarly, DTR indicates to the Data Set that the DTE is on. Data Carrier Detect (DCD) indicates that a good carrier is being received from the remote modem.

Pins 4 RTS (Request to Send - from the transmitting computer) and 5 CTS (Clear to Send - from the Data set) are used to control. In most Asynchronous situations, RTS and CTS are constantly on throughout the communication session. However where the DTE is connected to a multipoint line, RTS is used to turn carrier on the modem on and off. On a multipoint line, it's imperative that only one station is transmitting at a time (because they share the return phone pair). When a station wants to transmit, it raises RTS. The modem turns on carrier, typically waits a few milliseconds for carrier to stabilize, and then raises CTS. The DTE transmits when it sees CTS up. When the station has finished its transmission, it drops RTS and the modem drops CTS and carrier together.

Clock signals (pins 15, 17, & 24) are only used for synchronous communications. The modem or DSU extracts the clock from the data stream and provides a steady clock signal to the DTE.

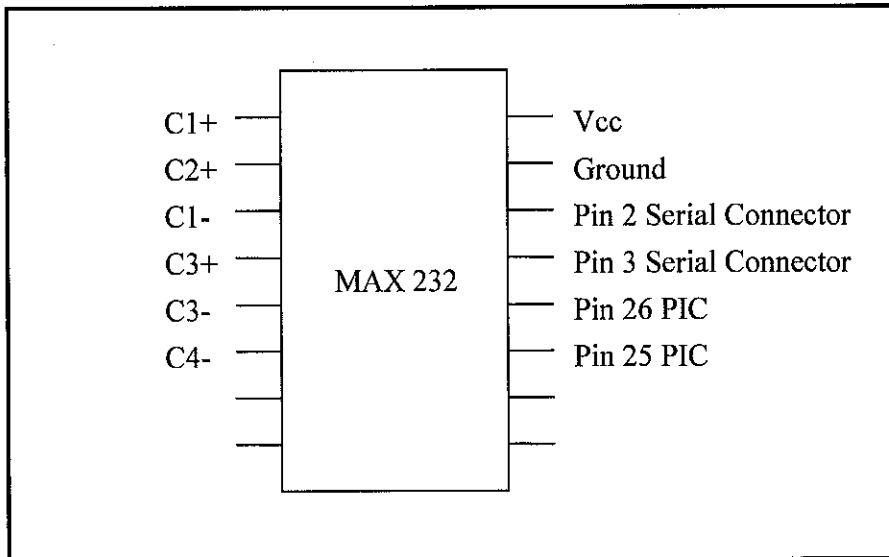


Figure 4.2: MAX232 configuration

For this project, only three pin is used which pin 2; received data, pin 3; transmitted data and pin 5; signal ground. These pin are connected to the MAX 232 through its pin 13 and 14. Four $1\mu\text{F}$ capacitor labeled C1 to C4 is used with MAX232.

4.4 GSM Modem

The GSM modem is the most important element in this project because the data transfer from one place to the terminal is done by the GSM modem using the GSM network. The hand phone is used as a GSM modem because its capability and features.

Most Nokia phones have F-Bus and M-Bus connections that can be used to connect a phone to a terminal or microcontroller. For this project, a Nokia 3310 is used. This is because the data cable for this model can be easily found and the model is quite cheap to purchase. The connection can be used for controlling just about all functions of the phone, as well as uploading new firmware etc. This bus will allow the user to send and receive SMS messages. The pin for the connection is located under the battery compartment. The four pin are labeled M-Bus/F-Bus, ground, receive and transmit.

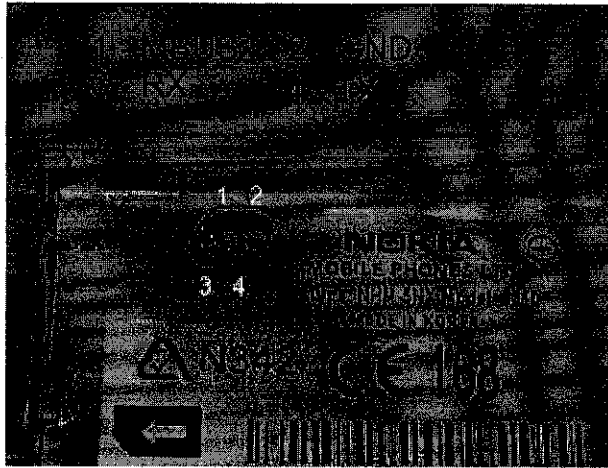


Figure 4.3: F-Bus/M-Bus Connection Pin in Nokia 3310

M-Bus is a one pin bi-directional bus for both transmitting and receiving data from the phone. It is slow (9600bps) and only half-duplex. Only two pins on the phone are used. One is ground pin and the other one is the data pin. M-Bus runs at 9600bps, 8 data bits, odd parity, and one stop bit. The data terminal ready (DTR) pin must be cleared with the request to send (RTS).

F-Bus is the later high-speed full-duplex bus. It uses one pin for transmitting data and one pin for receiving data plus the ground pin. Very much like a standard serial port. It is fast 115,200bps, 8 data bits, no parity, and one stop bit. For F-Bus the data terminal ready (DTR) pin must be set and the request to send (RTS) pin cleared. The serial cable contains electronics for level conversion and therefore requires power. The first thing to do is supply power to the cable electronics and this is done by setting the DTR (Data Terminal Ready) pin and clearing the RTS (Request to Send) pin. The next step is to synchronize the UART in the phone with your PC or microcontroller. This is done by sending a string of 0x55 or 'U' 128 times. The bus is now ready to be used for sending frames.

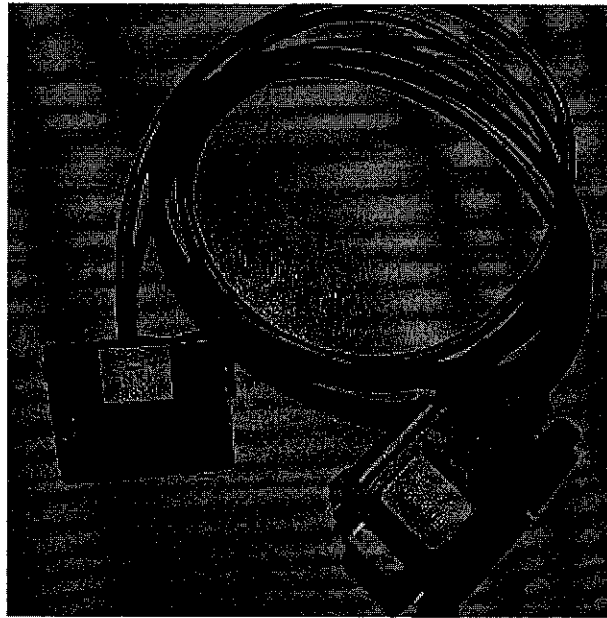


Figure 4.4: Data Cable for Nokia 3310

The Nokia protocol has a series of commands that allow the user to make calls, send and get SMS messages and lots more. The data cable is needed to connect the phone to the terminal via serial connector. Different models of phone require its own data cable. Figure below shows how the data cable is connected to Nokia 3310. After being connected to the terminal via serial port, the phone can be used as a GSM modem. The data received from the phone must be manipulated using software. In this project special designed software is used to manipulate the data received and sent by the user.



Figure 4.5: Data Cable Connection to Nokia 3310

4.5 Sending SMS

The standard SMS protocol is GSM 03.38 - Alphabets and language-specific information. This is the Technical Specification that describes the packing of 7-bit characters and shows the standard character map. For example, the string 'hello' is decoded. First, 'hello' must be displayed in hexadecimal using the character map provided in GSM 03.38. For A to Z and numbers it's just the standard ASCII conversion.

h	e	l	l	o	(ASCII characters)
68	65	6C	6C	6F	(In hexadecimal)
1101000	1100101	1101100	1101100	1101111	(In Binary)

When dealing with binary, it makes life easier to write everything backwards. The first byte in the string is on the right. The least significant bit is then displayed on the left with the most significant bit on the left. Shown below is the same string of 'hello' just displayed in reverse order. Then it's just a matter to dividing the binary values into bytes starting with the first character in the string. (Start from right and go to left.) The first decoded byte is simply the first 7 bits of the first character with the first bit of the second character added to the end as shown below. The next decoded byte is then the remaining 6 bits from the second character with two bits of the third byte added to the end. This process just keeps going until all characters are decoded. The last decoded byte is the remaining bits from the last character with the most significant bits packed with zeros.

6F	6C	6C	65	68	
1101111	1101100	1101100	1100101	1101000	(The ASCII characters shown in binary)
110	11111101	10011011	00110010	11101000	(The above binary just split into 8 bit segments)
06	FD	9B	32	E8	(The 8 bit segments decoded into hex)

The message hello is therefore E8 32 9B FD 06 when packed.

GSM 03.40 - Technical realization of the Short Message Service (SMS) Point-to-Point (PP). This specification describes the following SMS fields in detail.

Sample frame sent to Nokia 3310 (showed as a Hex dump) 98 Bytes

```
Byte: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20
Data: 1E 00 0C 02 00 59 00 01 00 01 02 00 07 91 16 14 91 09 10 F0 00

Byte: 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
Data: 00 00 00 15 00 00 00 33 0A 81 40 30 87 00 47 00 00 00 00 00 A7

Byte: 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62
Data: 00 00 00 00 00 00 C8 34 28 C8 66 BB 40 54 74 7A 0E 6A 97 E7 F3

Byte: 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
Data: F0 B9 0C BA 87 E7 A0 79 D9 4D 07 D1 D1 F2 77 FD 8C 06 19 5B C2

Byte: 84 85 86 87 88 89 90 91 92 93 94 95 96 97
Data: FA DC 05 1A BE DF EC 50 08 01 43 00 7A 52
```

F-Bus Frame Header

Byte 0: F-Bus Frame ID. Cable is 0x1E.

Byte 1: Destination address.

Byte 2: Source address.

Byte 3: Message Type or 'command'. 0x02 (SMS Handling).

Byte 4 & 5: Message length. In our case it is 0x0059 bytes long or 89 bytes in decimal.

(SMS) Short Message Service Frame Header

Byte 6 to 8: Start of the SMS Frame Header. 0x00, 0x01, 0x00

Byte 9 to 11: 0x01, 0x02, 0x00 = Send SMS Message

(SMSC) Short Message Service Centre (12 Bytes)

Byte 12: SMS Centre number length. 0x07 is 7 bytes long. This includes SMSC Number Type and SMS Centre Phone Number

Byte 13: SMSC number type e.g. 0x81-unknown 0x91-international 0xa1-national

1XXX IIII: Where I is the Numbering-plan-identification (Refer to GSM 03.40 - 9.1.2.5 Address fields)
1TTT XXXX: Where T is the Type-of-number (Refer to GSM 03.40 - 9.1.2.5 Address fields)

Byte 14 to 23: (Octet format) SMS Centre Phone Number In this case +61 411990010

(TPDU) Transfer Protocol Data Unit

Byte 24: Message Type

XXXX XXX1 = SMS Submit - The short message is transmitted from the Mobile Station (MS) to the Service Centre (SC).
XXXX XXX0 = SMS Deliver - The short message is transmitted from the SC to the MS.

(Refer to GSM 03.40 - 9.2.3 Definition of the TPDU parameters) In this case it is 0x15 = 0001 0101 in binary. The message is SMS Submit, Reject Duplicates, and Validity Indicator present.

Byte 25: Message Reference if SMS Deliver & Validity Indicator used (Not used in this case). Refer GSM 03.40 - 9.2.3.6 TP-Message-Reference (TP-MR)

Byte 26: Protocol ID. Refer to GSM 3.40 - 9.2.3.9 TP-Protocol-Identifier (TP-PID)

Byte 27: Data Coding Scheme. Refer to GSM 03.38 & GSM 3.40 - 9.2.3.10 TP-Data-Coding-Scheme (TP-DCS)

Byte 28: Message Size is 0x33 in hex or 51 bytes long in decimal. This is the size of the unpacked message.

Refer to GSM 03.40 - 9.2.3.16 TP-User-Data-Length (TP-UDL)

Destination's Phone Number (12 Bytes)

Byte 29: Destination's number length.

Byte 30: Number type e.g. 0x81-unknown 0x91-international 0xa1-national

Byte 31 to 40: (Octet format) Destination's Phone Number

Validity Period (VP)

Byte 41: Validity-Period Code. Time period during which the originator considers the short message to be valid.

Byte 42 to 47: Service Centre Time Stamp. For SMS-Deliver

The SMS Message (SMS-SUBMIT)

Byte 48 to 92: This is the SMS message packed into 7 bit characters. SMS Point-to-Point Character Packing

Byte 93: Always 0x00

The F-Bus Usual Ending

Byte 94: Packet Sequence Number

Byte 95: Padding Byte - String is odd and has to be even.

Byte 96 & 97: Odd & even checksum bytes.

4.6 Peripheral Interface Controller (PIC) PIC16F877

The PIC is used as the microcontroller in the system. It receives the data from the terminal and the devices and processes them. After processing the information, the required response is sent either to the terminal or to the devices.

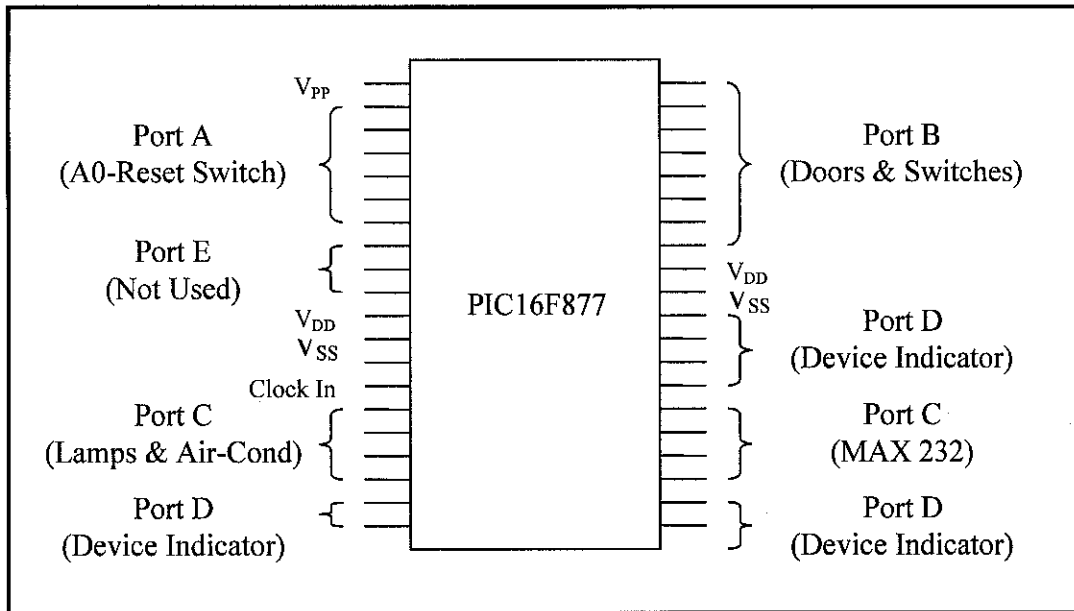


Figure 4.6: PIC Pin Configuration

As can be seen in figure 4.6, there are five input output port that can be used. In Port A, only one pin is used which is A0 as a reset switch for the PIC. The other pin is unused. Port B is used for the devices such as doors and switches for the lamps and air conditioner. Port C is used for the lamps and motor which represent air conditioner. Port D is used for LEDs that function as a status indicator for each device. Port E is still left unused. Therefore, there are still 14 pin left unused and a number of 7 more devices can be placed in the system. If the building requires controlling more devices, a larger capacity microcontroller has to be used. The PIC is programmable and C language is used to program it. The final program for the PIC can be seen in **Appendix A-13**.

4.7 Final Software- I-UTP Building Control v1.0

The final software is called I-UTP Building Control v1.0. The main window is divided into several parts which are the controller connection, F-Bus connection, switch panel, sensor panel, provider server information and the status log. Besides that, there are also buttons such as view terminal logs, view ASCII code, view serial information and close button.

The controller connection is used to connect the terminal with the microcontroller which is the PIC. When connected, the switch panel and the sensor panel will be on according to the current status given by the PIC. The F-Bus connection is used to connect the GSM modem to the terminal. When it is connected, the provider server information will be displayed. If the information is not displayed, there is a possibility that an error has occurred in the F-Bus connection.

The status log will show all the activities that occurred during the connection. The view terminal log button will direct the user to another window (**Appendix A-14**) which is mainly used for debugging purposes. At this window the real data that is sent and receive is shown. The view ASCII code is a window that shows the ASCII code and its equivalent conversion. This is very useful in the debugging process. The view serial information is just for the user to gain information about serial pin connection. The last button which is the close button will end the program.

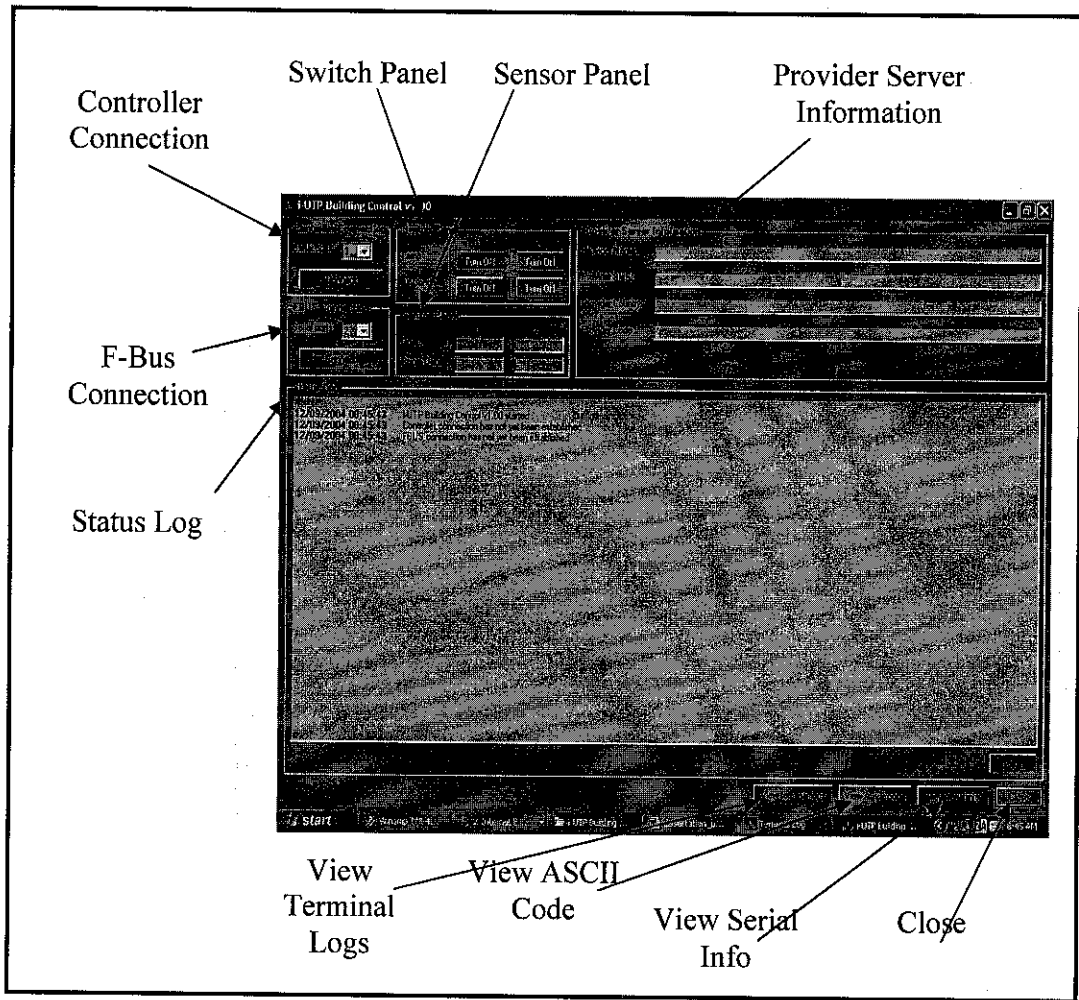


Figure 4.7: I-UTP Building Control Main Window

Section	Function
Controller Connection	To on/off the controller connection
F-Bus Connection	To on/off the F-Bus connection
Switch Panel	To manually on/off the devices
Status Panel	To show the current status of the monitored devices
Provider Server Information	Contain the necessary information of the server when connected
Status Log	Contain the log of all the activities done
View Terminal Logs	Direct the user to the Terminal Logs window
View ASCII code	Direct the user to the ASCII code window
View Serial Info	Direct the user to the Serial Info window
Close	To off the software

How to Use the System

1. Connect the serial cable and the F-Bus cable to the controller and mobile phone
2. Switch on the serial connection. Wait until the status log shows that the serial connection has been established.

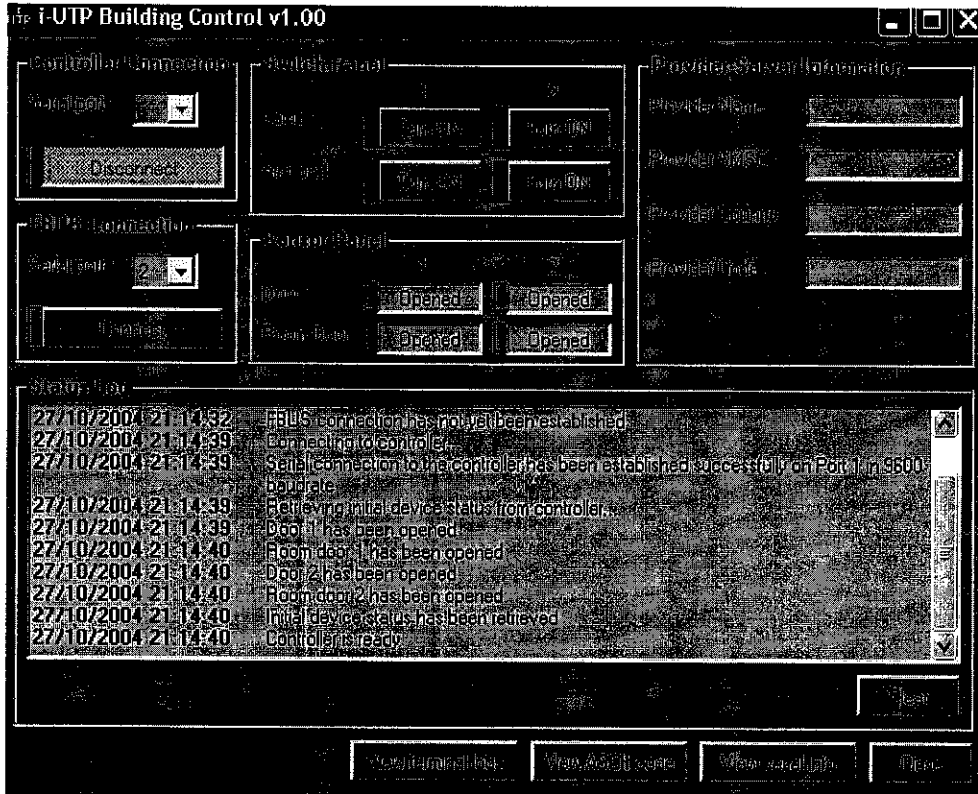


Figure 4.8: Serial Connection Established Successfully

3. Turn on the F-Bus connection on the software and waits for the F-Bus server to be ready. Type the required SMS message

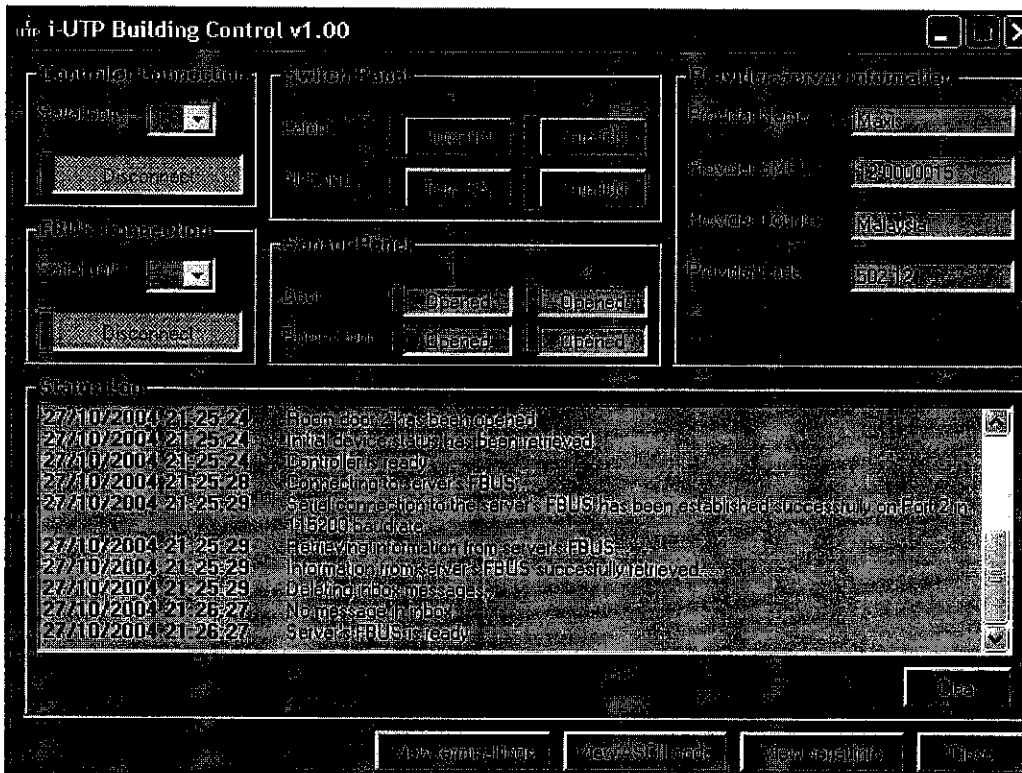


Figure 4.9: F-Bus Connection Established Successfully

4. Wait for the controller to respond and reply

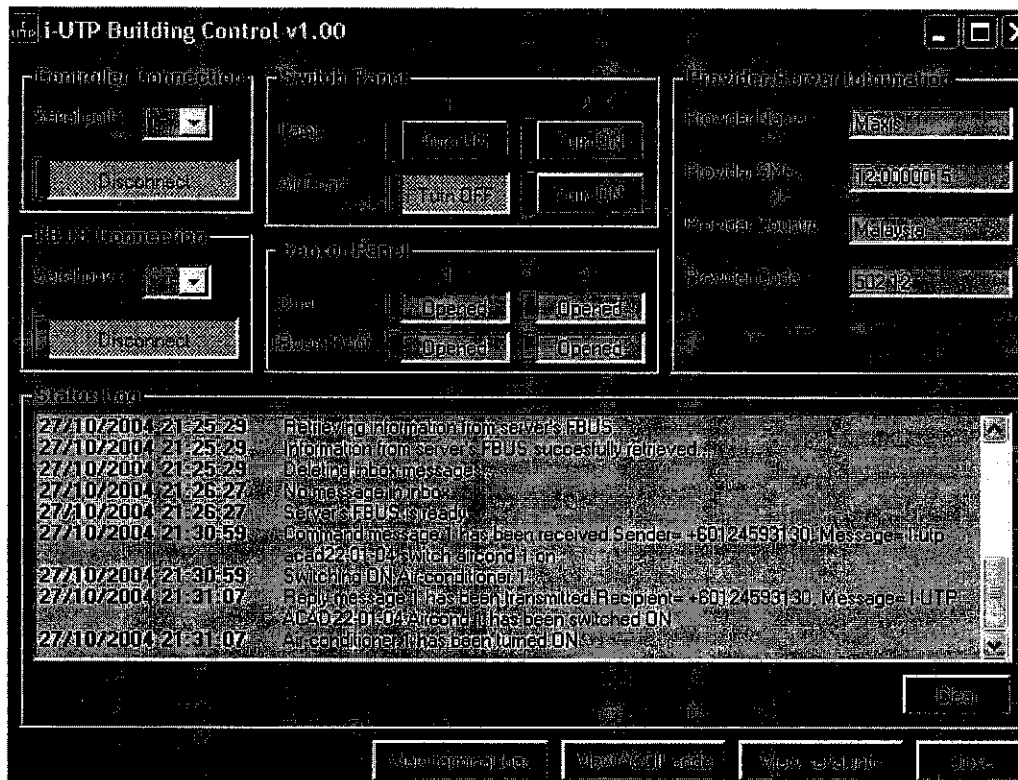


Figure 4.10: Message Receive and Reply Sent

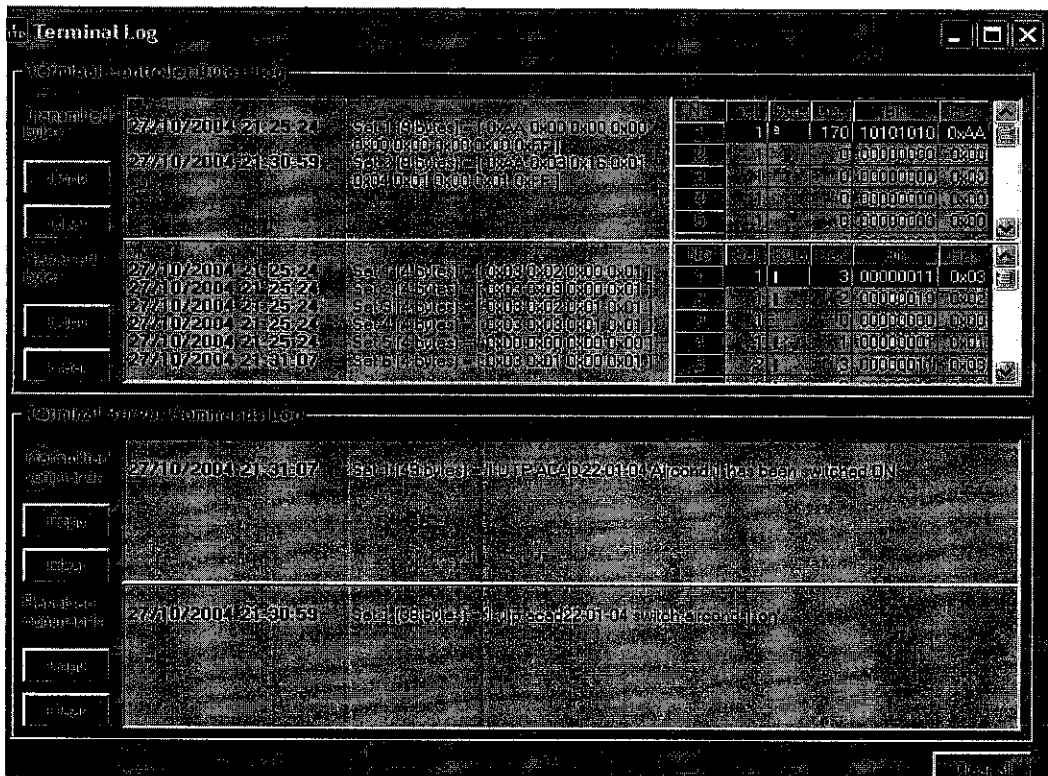


Figure 4.11: The Terminal Logs

4.8 Visual Basic

Visual Basic is a very important software in designing the system's software. Many of its function such as serial communication and F-Bus ActiveX Control are used. The section below discusses in detail what the functions that are used in developing the final software for this project.

4.8.1 Serial Connection

The serial connection software is used to test the serial connection whether the connection is functioning correctly or not. The following table lists the properties that are used to establish the serial connection:

Property Description

CommPort	Sets and returns the communications port number.
Settings	Sets and returns the baud rate, parity, data bits, and stop bits as a string.
PortOpen	Sets and returns the state of a communications port. Also opens and closes a port.

The sample screen of the software can be seen in **Appendix A-9**.

4.8.2 F-Bus ActiveX Control

Mobile FBUS 1.5 is a freeware ActiveX control that you can use to create software for mobile phones. Send SMS messages, manipulate operator logos, use monitoring, access phonebook, and much, much more. Mobile FBUS is the ideal tool for connecting your Visual Basic application to mobile phones. In this project, only the SMS message object is used.

This object can be used to read and send SMS messages. The SMS object has the following properties, method and sub-objects:

Property Description

LastError	Returns the last error code.
TotalMessages	Returns the total number of SMS messages stored.
UnreadMessages	Returns the number of unread messages.

Method Description

Refresh	Retrieves stored messages from the phone.
SendMessage	Sends an SMS message.

Object Description

DeliveryNotifications	Message box containing delivery notifications.
Inbox	Message box containing incoming messages.
Outbox	Message box containing sent messages.
FBSmsMessage	Sub-object representing one message in a message box.

Property Description

Count	Number of messages in box (read-only Integer). Messages have indexes between 1 and Count.
-------	--

Property Description

DateTime	Timestamp of message (read-only Date).
Destination	Destination of message (read-only String). In most phone types this property is only available in the OutBox.
LastError	Last FBUS error, 0 if last action was successful (read-only Integer).

Sender	Sender of message (read-only String). In most phone types this property is only available in the InBox.
SentRead	SentRead property returns true if the message is sent or read (read-only Boolean).
Text	Returns the message text (read-only String).

Method Description

Delete	Deletes the message from the phone memory. You have to call SMS.Refresh to update the state of the SMS object after deleting messages.
--------	--

4.9 Limitation of the System

Every system has its own limitation including this system. There are several limitations to this system. Some of it is beyond the designer's capability to avoid. Among them are:

- Number of devices that can be controlled or monitor
- Service provider connection

4.9.1 Number of devices

PIC16F877 is used in this project. This number of devices that the controller can handle problem occurs because the controller used is the PIC16F877. It only has five ports that could only accommodate 33 I/O. This problem can be overcome by using a bigger capacity controller or placing one microcontroller for each building level. Although the cost might increase, this does not cause a very significant impact because the cost of this system is not very expensive. Another approach to overcome this problem is to install this system can at each building level.

4.9.2 Service Provider Connection

The system also depends on the connection that the service provider provides. If the connection is poor, there is a chance that the message send will be delayed or not received by the gsm modem. This will result in the system not functioning correctly. This is also not a very big problem because the services offers now are very efficient because there are competitions between the providers. Because of this healthy competition, the services quality is increasing by each passing day.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

This chapter is divided into two topics which is the conclusion and the recommendation. The conclusion part discussed about what the project's objectives and how the project is completed. The recommendation part discussed about the future improvement of the project and in what other area the project can be implemented.

5.1 CONCLUSION

In conclusion, this project is very useful to understand in-depth about Nokia F-Bus protocol and its application features. The knowledge gained is very helpful to identify and understand all other communication protocols generally.

In order to complete this project, knowledge about F-Bus protocols and PIC programming is very important. A lot of research is done first before the real implementation can be done. The system is hoped to help make the monitoring and controlling a building a much easier task. Using this system, the worker does not have to go up the building to check each device. They only have to do such things only when there is an emergency or device breakdown happen.

The communication using GSM and the services it offers nowadays are becoming more popular and can be utilize further more with new innovation and integrating the system with other existing system to make them more efficient and better. This project is only one way of making the full use of the GSM network.

The objectives of this project have been fulfilled and the project is hoped to make UTP building a more cost effective and easy to monitor building.

5.2 RECOMMENDATIONS

Despite of the success of the project, there are still rooms for improvements. For example, the delay the system takes to initialize especially the F-Bus connection can be reduced. This problem arise because of the system needs to refresh every 60 seconds to check for incoming messages. This time can be reduced to minimize the delay.

Other recommendation is to use a higher power PIC to accommodate more devices. The PIC16F877 can only accommodate 11 devices, this is not enough to control and monitor a building with a lot of devices such as computers and machinery.

The system can also be expanded to make it have more functions such as integrate it with voice recognition. The users just have to call the server and tell the controller what to do by using some specific commands.

Other than using the SMS to control devices or building, it can also be used to keep track of vehicle. This can be done by integrating the SMS with the Global Positioning System (GPS) system. The system can also be integrated with the Geographical Information System (GIS) System to make it easier for the driver to find certain route to the destination.

There are also on going research about the usage of SMS service in other system such as SMS Car Parking Payment, SMS Bill Reminder Payment Systems, Machine-to-Machine (M2M); Mobile-to-machine and Machine-to-mobile, Home Surveillance with Mobile Phones and Mobile Phone Based Ticketing (transportation, train, parking meters etc)

REFERENCES

- [1] <http://www.comms.eee.strath.ac.uk/~gozalvez/gsm/gsm.html>, *An Overview of the GSM System* by **Javier Gozálvéz Sempere**
- [2] <http://kbs.cs.tu-berlin.de/~jutta/gsm/js-intro.html>, *A Brief Overview of GSM*, by **John Scourias**
- [3] <http://www.embedtronics.com/nokia/fbus.html#part1>, *Nokia F-Bus Protocol Made Simple* by **Wayne Peacock**
- [4] **William Stallings**, *Data and Computer Network*, 7th Edition, Prentice Hall 2004
- [5] **Dr P.Sellapan**, *Visual Basic through Examples*, 1st Edition, Federal Publication Sdn. Bhd. 2003
- [6] <http://www.ihub.com/GSM%20Modems.htm>, *GSM Modems*
- [7] <http://www.handytel.com/technology/gsm03.htm>, *GSM System Architecture*
- [8] <http://softwarecaves.com>, *MFBUS 1.5 ActiveX Control*
- [9] <http://gnokii.org>, *Linux Gnokii Projects*
- [10] <http://www.weethet.nl/>, *Pin Layout for Nokia 3310*
- [11] <http://www.activexperts.com/activcomport>, *Nokia GSM AT Commands*
- [12] *PIC16F877 Data Sheet, 28/40 Pin 8-Bit CMOS FLASH Microcontrollers* by **Microchip**

APPENDICES

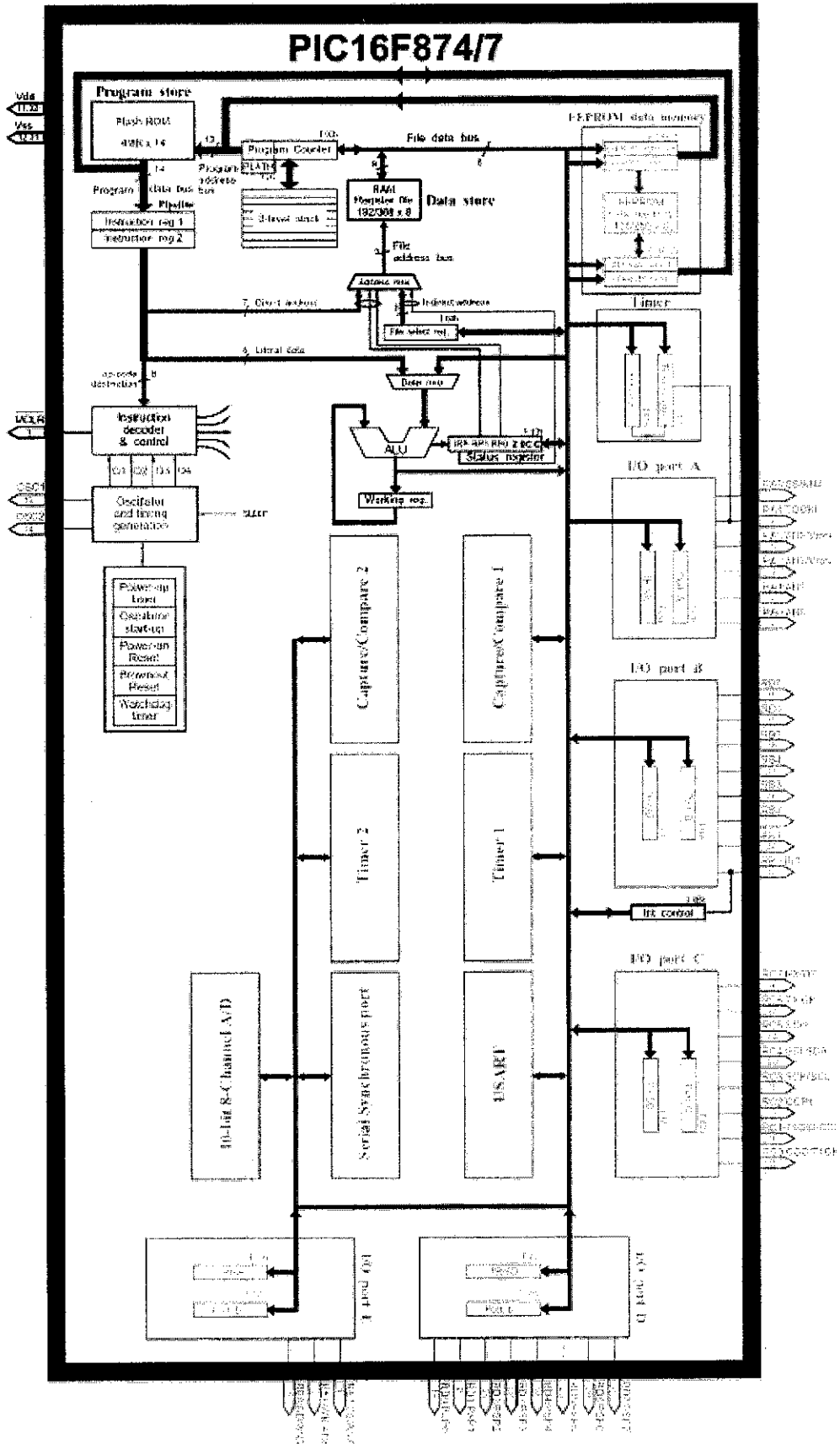


Figure A-1: Architecture of the PIC16F877 microcontroller

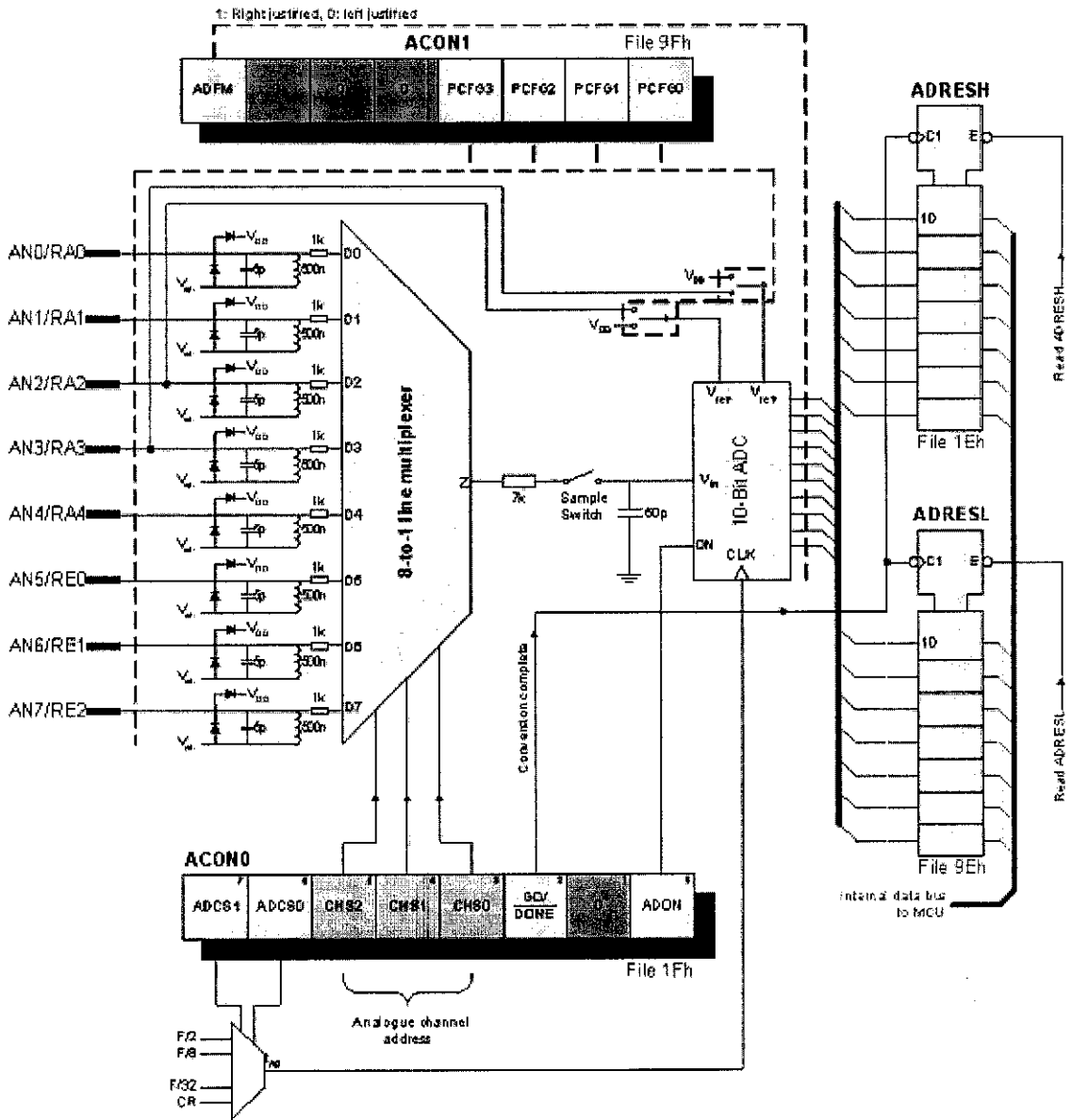


Figure A-2: Simplified block diagram of the PIC16F877 ADC module

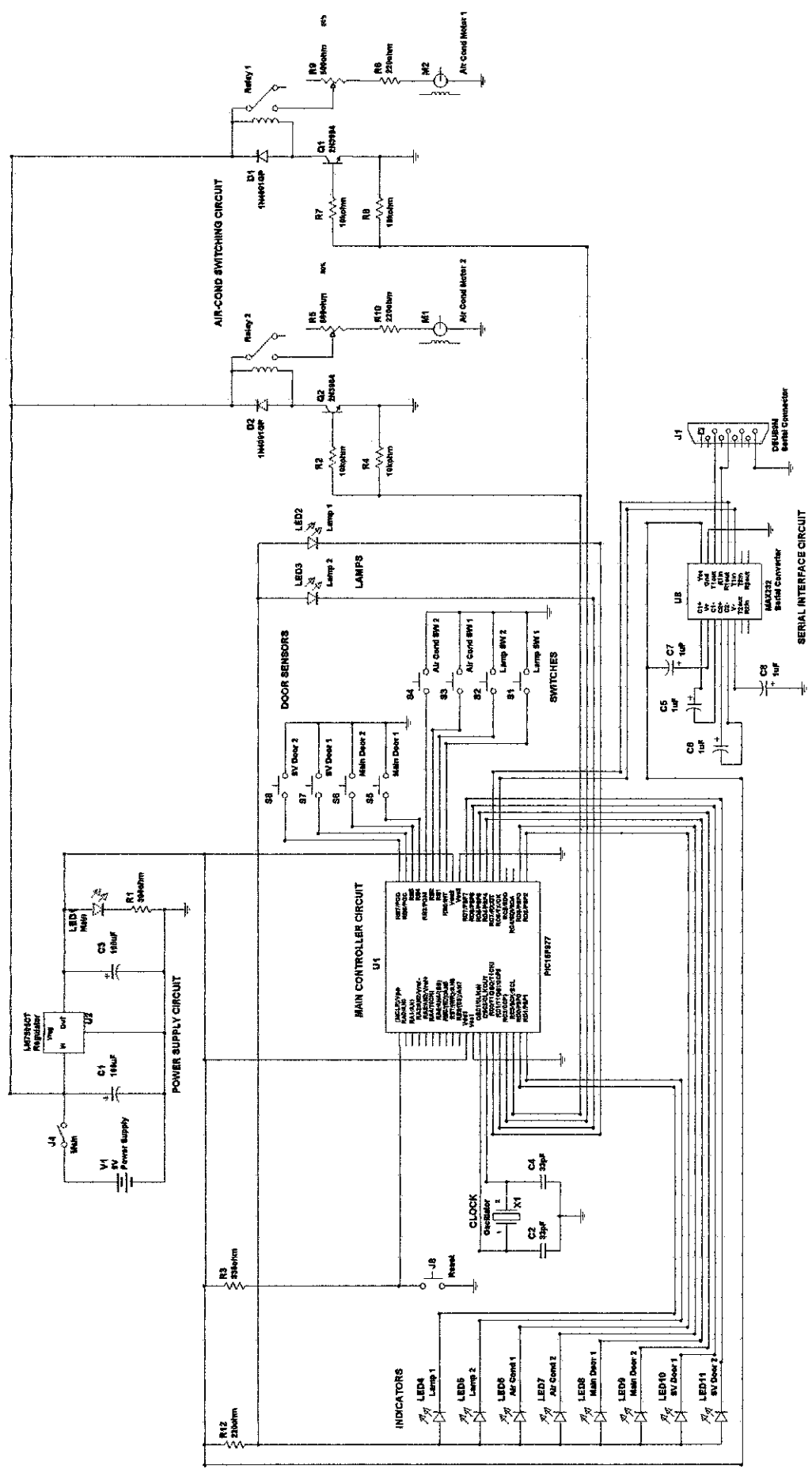


Figure A-4: Hardware Circuit

Title: Final Circuit	
FR013	
Designed by:	Maria Saadik
Checked by:	Ahmed Hammad
Approved by:	Mr. Azman
Document No.	0401
Date	June 19, 2004
Sheet	1 of 1
Revision	1.0
Drawn	AE

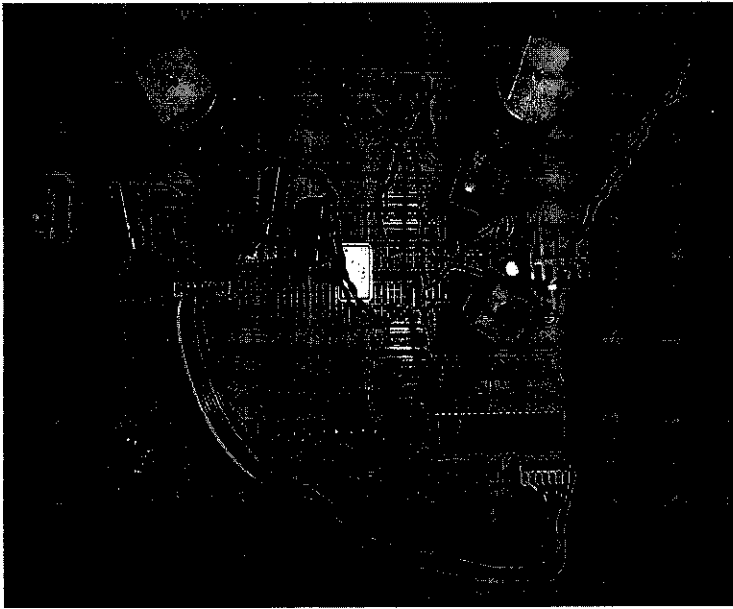


Figure A-5: Hardware Test Board

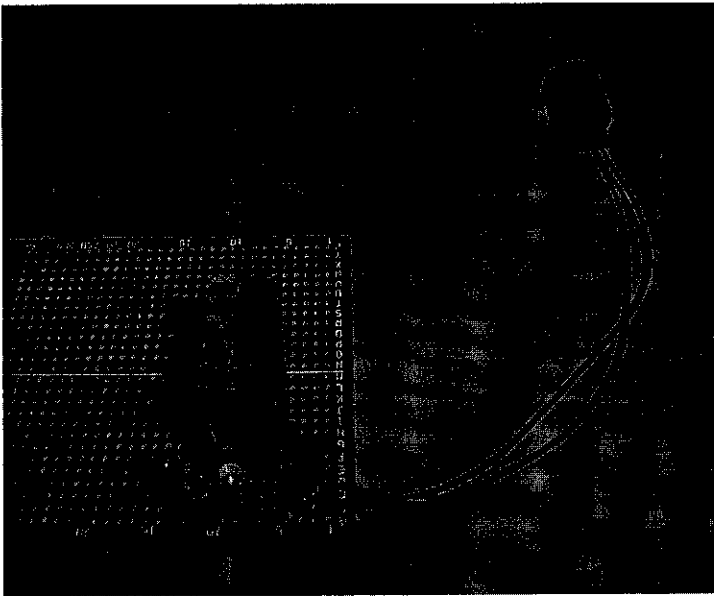
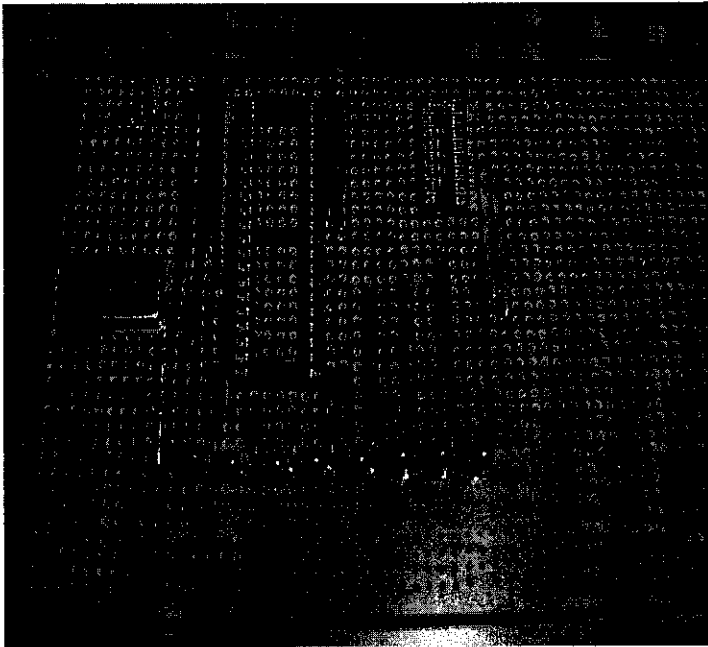


Figure A-6: Hardware Implementation on Vero Board

```

#include <16F877.H>
#include XT_NOPROTECT,NOWDT,NOLVP
#include delay(clock=4000000)

#include <conio.h>
#include <stdio.h>

#define bldata 27

void main()
{
    int buffer1;
    int count1;
    int input_data_even[bldata];
    int input_data_odd[bldata];

    int checksum_result_even;
    int checksum_result_odd;

//All *even
input_data_even[0]=0x03;
input_data_even[1]=0x20;
input_data_even[2]=0x34;
input_data_even[3]=0x34;
input_data_even[4]=0x0A;
input_data_even[5]=0x31;
input_data_even[6]=0x30;
input_data_even[7]=0x2D;
input_data_even[8]=0x31;
input_data_even[9]=0x4E;
input_data_even[10]=0x4D;
input_data_even[11]=0x35;
input_data_even[12]=0x28;
input_data_even[13]=0x29;
input_data_even[14]=0x4E;
input_data_even[15]=0x50;
input_data_even[16]=0x00;
input_data_even[17]=0x47;

input_data_even[18]=0x0C;
input_data_even[19]=0x7F;
input_data_even[20]=0x02;
input_data_even[21]=0x00;
input_data_even[22]=0x71;
input_data_even[23]=0x0C;
input_data_even[24]=0xD2;
input_data_even[25]=0x26;

input_data_even[26]=0x00;

//All odd
input_data_odd[0]=0x00;
input_data_odd[1]=0x56;
input_data_odd[2]=0x30;
input_data_odd[3]=0x2E;
input_data_odd[4]=0x35;
input_data_odd[5]=0x32;
input_data_odd[6]=0x2D;
input_data_odd[7]=0x36;
input_data_odd[8]=0x30;
input_data_odd[9]=0x0A;
input_data_odd[10]=0x48;
input_data_odd[11]=0x2D;
input_data_odd[12]=0x0A;

```

```

input_data_odd[13]=0x63;
input_data_odd[14]=0x20;
input_data_odd[15]=0x4D;
input_data_odd[16]=0x2E;
input_data_odd[17]=0x01;

input_data_odd[18]=0x1E;
input_data_odd[19]=0x00;
input_data_odd[20]=0x00;
input_data_odd[21]=0xD1;
input_data_odd[22]=0xCF;
input_data_odd[23]=0x1E;
input_data_odd[24]=0x00;
input_data_odd[25]=0x00;

input_data_odd[26]=0x01;

//Start XOR even
checksum_result_even=input_data_even[0];
for (count1=1;count1<=(bldata-1);count1++)
    {
        buffer1=input_data_even[count1];
        checksum_result_even=checksum_result_even^buffer1;
    }

//Start XOR odd
checksum_result_odd=input_data_odd[0];
for (count1=1;count1<=(bldata-1);count1++)
    {
        buffer1=input_data_odd[count1];
        checksum_result_odd=checksum_result_odd^buffer1;
    }

//Odd, Even

printf("Checksum = %X %X",checksum_result_odd,checksum_result_even);
getch();
}

```

Figure A-7: Checksum Program


```

#include <16F877.H>
#include XT,NOPROTECT,NOWDT,NOLVP
#include delay(clock=4000000)

#include <iostream.h>
#include <conio.h>
#include <stdio.h>
#include <math.h>
#define  bildata  40

//Debug
void DEC_BIN(int choose, int input);

void main()
{
    int input_data[bildata];
    int buffer_array[bildata];

    //for packing
    int count1;
    int count2;
    int buffer1;
    int buffer2;
    int buffer3;
    int buffer4;
    int buffer5;
    int shift1;
    int minus_array;
    int minus_shift;
    int had_recursion;

    //Swap from front to back, put into buffer
    for (count1=0;count1<=(bildata-1);count1++)
    {
        buffer_array[(bildata-1)-count1]=input_data[count1];
    }

    //Mask with 01111111 (to confirm value of 7 bits)
    for (count1=0;count1<=(bildata-1);count1++)
    {
        buffer5=buffer_array[count1];
        buffer5=buffer5&0x7F;
        buffer_array[count1]=buffer5;
    }

    //Pack septet to be octet

    //minus_array=bildata/8; //will be used
    //had_recursion=bildata-minus_array; //will be used
    minus_shift=1;

    for (count1=1;count1<=(bildata-1);count1++)
    {
        //Restart all 8 bytes
        if (((count1-1)%7==0) && (count1>1))
        {
            //Pull all value to the front (one byte step)
            for (count2=bildata-count1;count2>=1;count2--)
            {
                buffer_array[count2]=buffer_array[count2-1];
            }
            if (count2==1)
            {
                buffer_array[count2-1]=0x00;
            }
        }
        //Reset shift counter
        minus_shift=1;
    }
}

```

```

//Temporarily hold current value
buffer1=buffer_array[(bldata-count1)-1];
buffer2=buffer_array[(bldata)-count1];
//Shift previous value to the left
shift1=8-minus_shift;
buffer3=buffer1<<shift1;

//Debug
buffer3=buffer3&0xFF;

//Insert new value to the current value
buffer3=buffer3|buffer2;
buffer_array[(bldata)-count1]=buffer3;

//Shift previous value to the right
//Insert new value to the current previous value
buffer4=buffer1>>minus_shift;
buffer_array[(bldata-count1)-1]=buffer4;

//Increase shift counter
++minus_shift;

//Debug
for (int counttest=0;counttest<=(bldata-1);counttest++)
{
//Dec-Bin
        int digit,power=0,output=0;
        int buffer=buffer_array[counttest];
        while (buffer>0)
        {
                digit=buffer%2;
                output=output+digit*pow(10,power);
                buffer=buffer/2;
                ++power;
        }
        printf("%X = %d\t ",buffer_array[counttest],output);
}
printf("\n\n\n");
getch();
}

//Debug
printf("end");
getch();
}

/*
        int digit,power=0,output=0;
        int buffer=input;
        while (buffer>0)
        {
                digit=buffer%10;
                buffer=buffer/10;
                output=output+pow(2,power)*digit;
                ++power;
        }
*/

```

Figure A-8: Packer Program

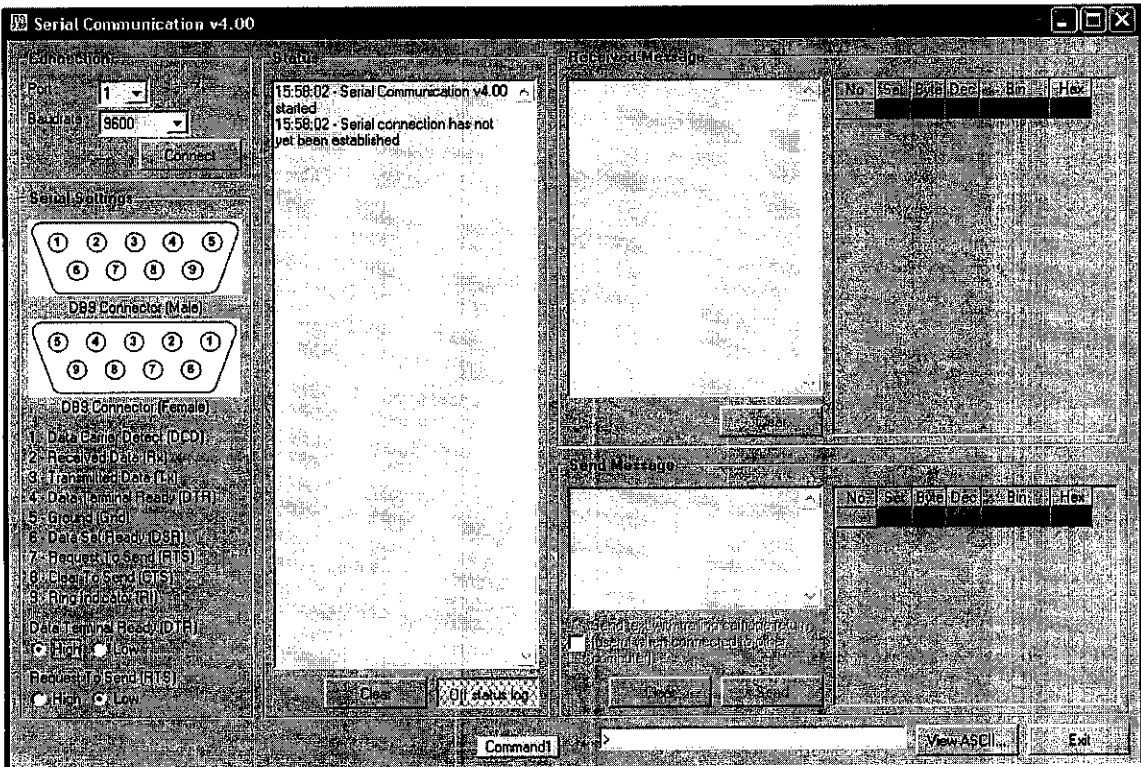


Figure A-9: Serial Interface Program

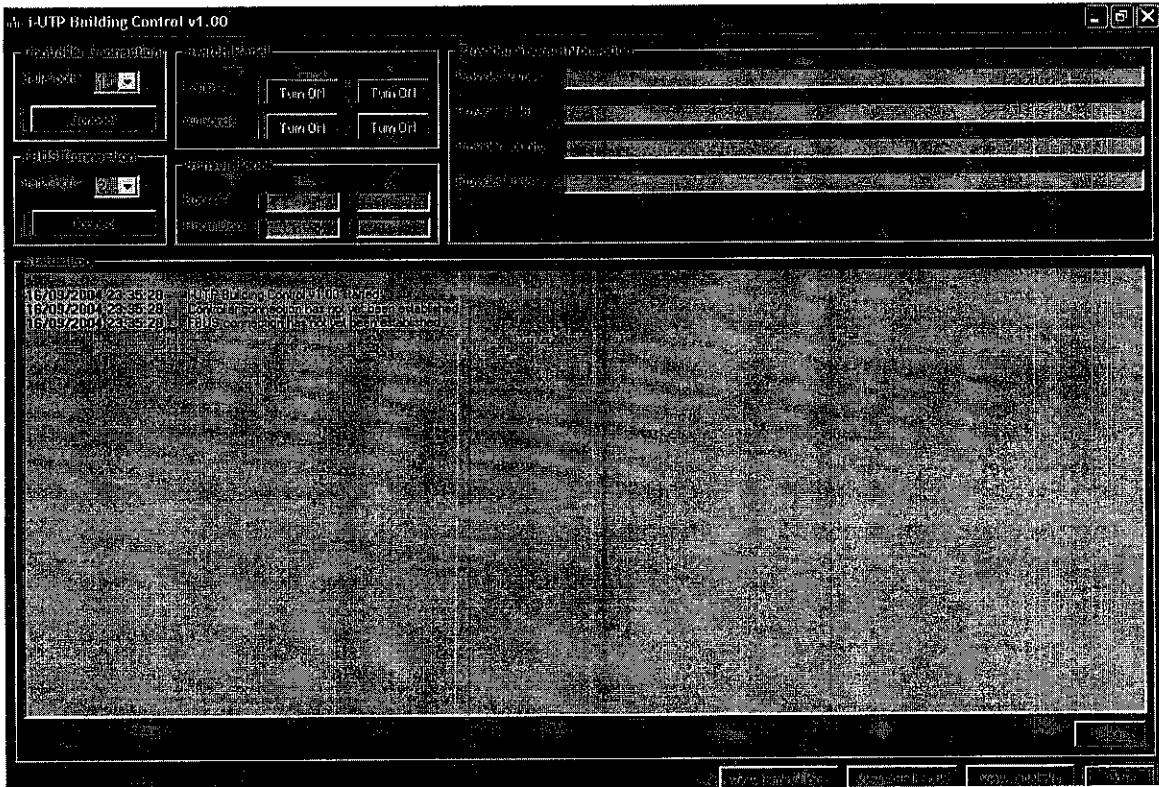


Figure A-10: Main Page for I-UTP Building Control

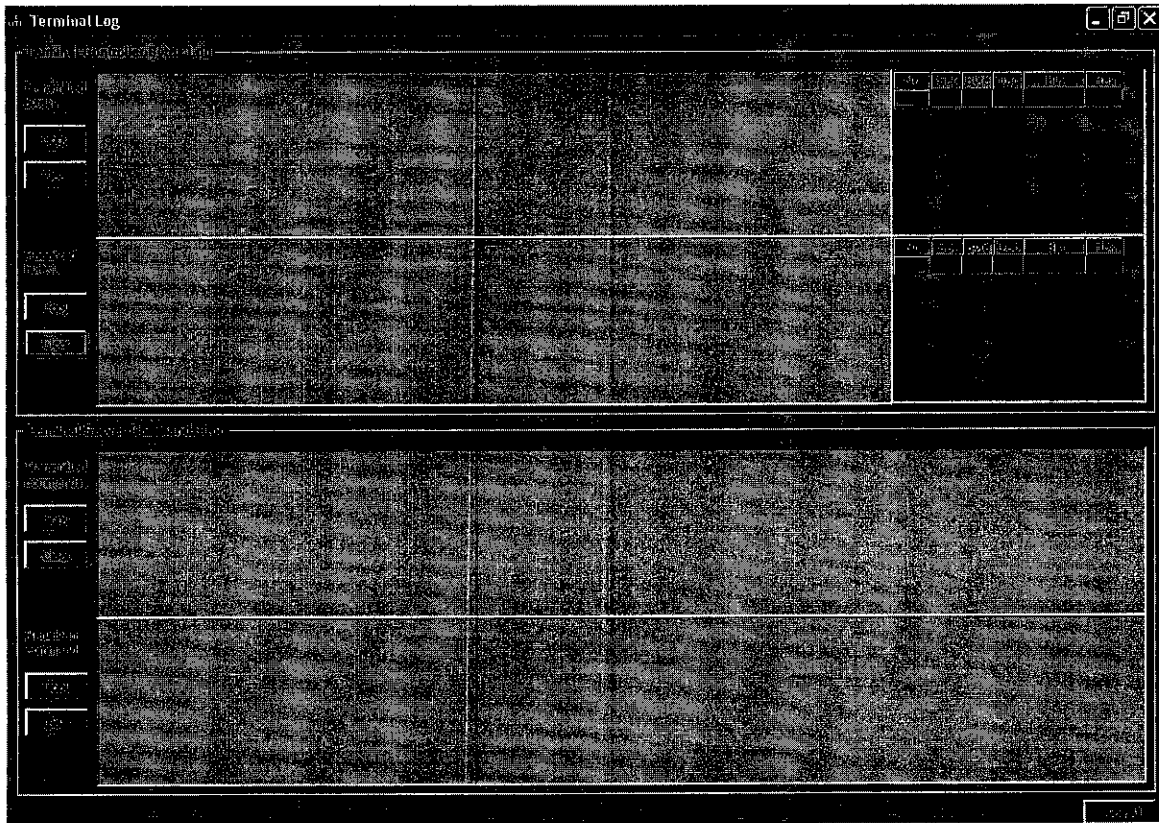


Figure A-11: Terminal Log for I-UTP Building Control

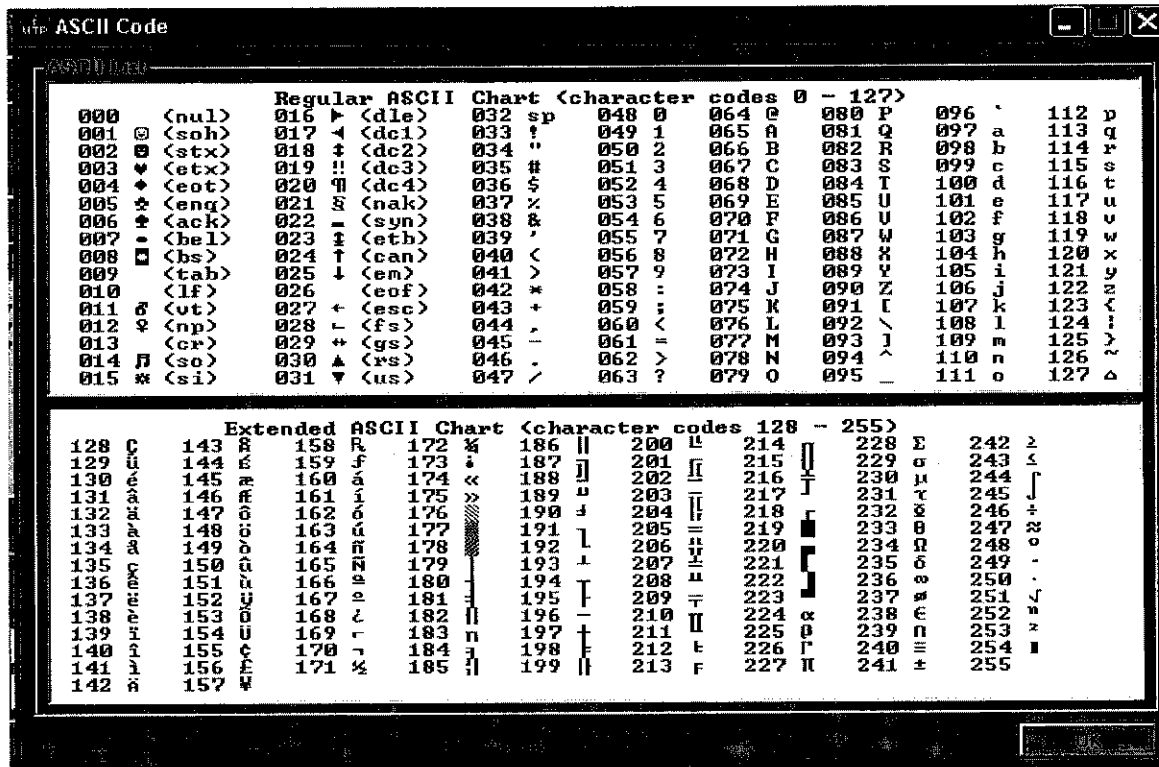


Figure A-12: ASCII Code Window

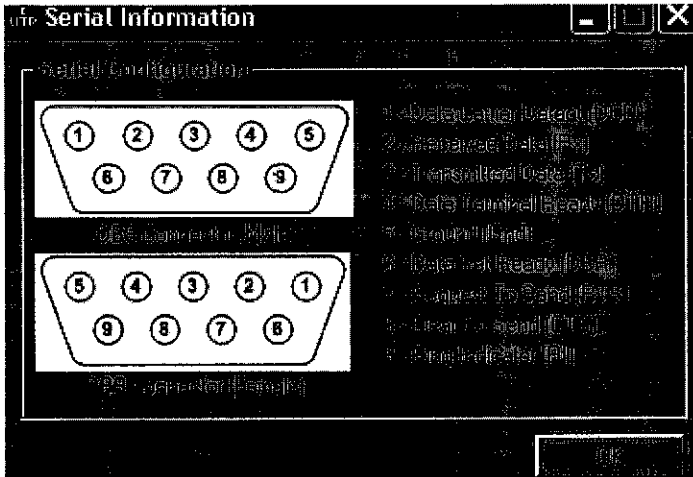


Figure A-13: Serial Information Window

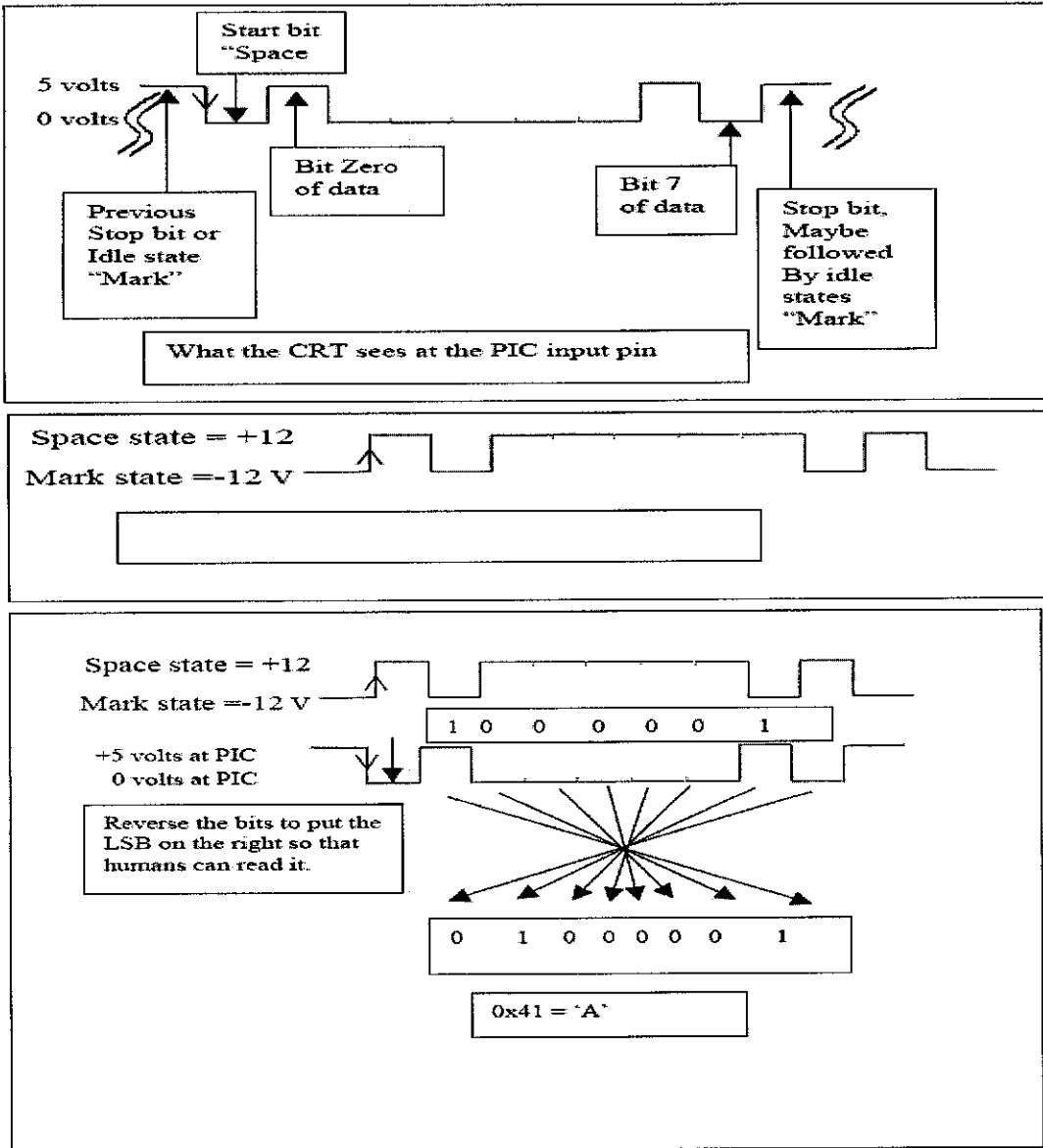


Figure A-14: How data is transferred in MAX232 and displayed in Oscilloscope

PIC Program

```

/*
A0 =
A1 =
A2 =
A3 =
A4 =
A5 =

B0 = Lamp 1 Switch (I)
B1 = Lamp 2 Switch (I)
B2 = Aircond 1 Switch (I)
B3 = Aircond 2 Switch (I)
B4 = Main Door 1 Sensor (I)
B5 = Main Door 2 Sensor (I)
B6 = Room Door 1 Sensor (I)
B7 = Room Door 2 Sensor (I)

C0 = Lamp 1 (O)
C1 = Lamp 2 (O)
C2 = Aircond 1 (O)
C3 = Aircond 1 (O)
C4 =
C5 =
C6 = Serial Transmit (O)
C7 = Serial Receive (I)

D0 = Lamp 1 Indicator (O)
D1 = Lamp 2 Indicator (O)
D2 = Aircond 1 Indicator (O)
D3 = Aircond 2 Indicator (O)
D4 = Main Door 1 Indicator (O)
D5 = Main Door 2 Indicator (O)
D6 = Room Door 1 Indicator (O)
D7 = Room Door 2 Indicator (O)

E0 =
E1 =
E2 =
*/

#include <16F877.H>
#fuses XT,NOWDT,NOPROTECT,NOLVP

#use delay(clock=4000000)
#use rs232(baud=9600, xmit=PIN_C6, rcv=PIN_C7)

byte data_counter=0;
byte data_status=0; //0=No transmission, 1=Data completed succesfully, 2=Data error

//All commands data
byte frame_start_byte=0;
byte frame_cmd_grp=0;
byte frame_bldg_num=0;
byte frame_flr_num=0;
byte frame_room_num=0;
byte frame_dev_num=0;
byte frame_dev_ID_num=0;
byte frame_dev_onoff=0;
byte frame_stop_byte=0;

//Device Status for each device
#define device_qty2
byte lamp[device_qty];
byte aircond[device_qty];
byte door[device_qty];
byte roomdoor[device_qty];
//ON/OFF Switch Status for each device
//(to prevent continuously ON when user press switch)

```

```

byte lamp_switch[device_qty];
byte aircond_switch[device_qty];

//Switch function
void SWITCH_COMMAND(int device_type,int device_no,int onoff_status);
void REPLY_INITIALIZATION(void);

//Interrupt for serial on-received
#int_rda
void serial_interrupt()
{
    int data_rcv;

    data_rcv=getch();

    //Check for start byte
    frame_start_byte=data_rcv;
    if (frame_start_byte==0xAA)
    {
        //Reset data counter
        data_counter=0;
        //Reset data status
        data_status=1;
    }
    else
    {
        switch(data_counter)
        {
            //Check for command group
            case 1:
            {
                if (data_status==1)
                {
                    frame_cmd_grp=data_rcv;
                    switch(frame_cmd_grp)
                    {
                        case 0x00:
                        case 0x01:
                        case 0x02:
                        case 0x03:
                        {
                            data_status=1;
                            break;
                        }
                        default:
                        {
                            data_status=2;
                            break;
                        }
                    }
                }
            }
            break;

            //Check for building number
            case 2:
            {
                if (data_status==1)
                {
                    frame_bldg_num=data_rcv;
                    switch(frame_bldg_num)
                    {
                        case 0x00:
                        case 0x16:
                        {
                            data_status=1;

```

```

        break;
    }
default:
    {
data_status=2;
        break;
    }
}

break;
}

//Check for floor number
case 3:
    {
        if (data_status==1)
        {
            frame_flr_num=data_rcv;
            switch(frame_flr_num)
            {
                case 0x00:
                case 0x01:
            {
data_status=1;
                    break;
                }
            default:
            {
data_status=2;
                    break;
                }
            }
        }
    }

break;
}

//Check for room number
case 4:
    {
        if (data_status==1)
        {
            frame_room_num=data_rcv;
            switch(frame_room_num)
            {
                case 0x00:
                case 0x04:
            {
data_status=1;
                    break;
                }
            default:
            {
data_status=2;
                    break;
                }
            }
        }
    }

break;
}

//Check for device number
case 5:
    {
        if (data_status==1)
        {
            frame_dev_num=data_rcv;
            switch(frame_dev_num)

```



```

{
case 0x00:
case 0x01:
{
data_status=1;
break;
}
default:
{
data_status=2;
break;
}
}

break;
}

//Check for device ID number
case 6:
{
if (data_status==1)
{
frame_dev_ID_num=data_rcv;
switch(frame_dev_ID_num)
{
case 0x00:
case 0x01:
{
data_status=1;
break;
}
default:
{
data_status=2;
break;
}
}
}
}

break;
}

//Check for device on/off command
case 7:
{
if (data_status==1)
{
frame_dev_onoff=data_rcv;
switch(frame_dev_onoff)
{
case 0x00:
case 0x01:
{
data_status=1;
break;
}
default:
{
data_status=2;
break;
}
}
}
}

break;
}

//Check for stop byte
case 8:

```

```

        {
            if(data_status==1)
            {
                frame_stop_byte=data_rcv;

                switch(frame_stop_byte)
                {
                    case 0xFF:
                    {
                        data_status=1;
                        break;
                    }
                    default:
                    {
                        data_status=2;
                        break;
                    }
                }
            }

            break;
        }
    }

    if(data_counter==8)
    {
//If all commands are correct
//-----
        if(data_status==1)
        {
//Execute command!!
//-----
            //Check for command group
            if(frame_cmd_grp==0x03)
            {
                //Check for building number
                if(frame_bldg_num==0x16)
                {
                    //Check for floor number
                    if(frame_flr_num==0x01)
                    {
                        //Check for room number
                        if(frame_room_num==0x04)
                        {
                            //Check for device number
                            //LAMP
                            if(frame_dev_num==0x00)
                            {
                                //Check for device ID number
                                //LAMP1
                                if(frame_dev_ID_num==0x00)
                                {
                                    //OFF
                                    if(frame_dev_onoff==0x00)
                                    {
                                        //Turn OFF

                                        SWITCH_COMMAND(0,0,0);
                                    }
                                    else
                                    //ON
                                    if(frame_dev_onoff==0x01)
                                    {
                                        //Turn ON

                                        SWITCH_COMMAND(0,0,1);
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

    }
  }
  else
    //LAMP2
    if (frame_dev_ID_num==0x01)
    {
      //OFF
      if (frame_dev_onoff==0x00)
      {
        //Turn OFF

SWITCH_COMMAND(0,1,0);
      }
      else
        //ON
        if (frame_dev_onoff==0x01)
        {
          //Turn ON

SWITCH_COMMAND(0,1,1);
        }
      }
    }
  else
    //AIRCOND
    if (frame_dev_num==0x01)
    {
      //Check for device ID number
      //AIRCOND1
      if (frame_dev_ID_num==0x00)
      {
        //OFF
        if (frame_dev_onoff==0x00)
        {
          //Turn OFF

SWITCH_COMMAND(1,0,0);
        }
        else
          //ON
          if (frame_dev_onoff==0x01)
          {
            //Turn ON

SWITCH_COMMAND(1,0,1);
          }
        }
      }
    }
  else
    //AIRCOND2
    if (frame_dev_ID_num==0x01)
    {
      //OFF
      if (frame_dev_onoff==0x00)
      {
        //Turn OFF

SWITCH_COMMAND(1,1,0);
      }
      else
        //ON
        if (frame_dev_onoff==0x01)
        {
          //Turn ON

SWITCH_COMMAND(1,1,1);
        }
      }
    }
  }
}

```

```

    }
    }
}
else
//If it is initialization frame
if (frame_cmd_grp==0x00 && frame_bldg_num==0x00 && frame_flr_num==0x00 && frame_room_num==0x00 &&
frame_dev_num==0x00 && frame_dev_ID_num==0x00 && frame_dev_onoff==0x00)
{
    REPLY_INITIALIZATION();
}
}

//Reset counter n status
data_counter=0;
data_status=0;
}
else
{
    //Increment counter
    data_counter=data_counter+1;
}
}

void main()
{
    int count1;
    int device_ID;

    //Enable interrupt for serial on-received
    enable_interrupts(global);
    enable_interrupts(int_rda);
    //Enable weak internal pull-up resistor on Port B
    port_b_pullups(true);

    //Output Initialization
    //--Indicators--
    output_bit(PIN_D0,1);
    output_bit(PIN_D1,1);
    output_bit(PIN_D2,1);
    output_bit(PIN_D3,1);
    output_bit(PIN_D4,1);
    output_bit(PIN_D5,1);
    output_bit(PIN_D6,1);
    output_bit(PIN_D7,1);
    //--Lamps--
    output_bit(PIN_C0,1);
    output_bit(PIN_C1,1);
    //--Airconds--
    output_bit(PIN_C2,0);
    output_bit(PIN_C3,0);

    //Status Initialization
    //--Lamps & Airconds--
    for (count1=0;count1<=device_qty-1;count1++)
    {
        //Device Status
        //0=OFF, 1=ON
        //(Default=0)
        lamp[count1]=0;
        aircond[count1]=0;
        door[count1]=0;
        roomdoor[count1]=0;
        //ON/OFF Switch Status
        //0=OFF, 1=ON
        //(Default=1)
        lamp_switch[count1]=1;
        aircond_switch[count1]=1;
    }
}

```

```

while(true)
{
    //----SWITCH DETECTION----

    //*****LAMP 1*****
    //If the switch is pressed
    device_ID=0;
    if (input(PIN_B0)==0)
    {
        //If the switch has been released previously
        if (lamp_switch[device_ID]==1)
        {
            //If the device is currently OFF
            if (lamp[device_ID]==0)
            {
                //Turn ON
                SWITCH_COMMAND(0,device_ID,1);
            }
        }
        else
        {
            //If the device is currently ON
            if (lamp[device_ID]==1)
            {
                //Turn OFF
                SWITCH_COMMAND(0,device_ID,0);
            }
        }
        //Change switch status
        lamp_switch[device_ID]=0;
    }
}
else
{
    //If its switch is released
    if (input(PIN_B0)==1)
    {
        //Change switch status
        lamp_switch[device_ID]=1;
    }
}

    //*****LAMP 2*****
    //If the switch is pressed
    device_ID=1;
    if (input(PIN_B1)==0)
    {
        //If the switch has been released previously
        if (lamp_switch[device_ID]==1)
        {
            //If the device is currently OFF
            if (lamp[device_ID]==0)
            {
                //Turn ON
                SWITCH_COMMAND(0,device_ID,1);
            }
        }
        else
        {
            //If the device is currently ON
            if (lamp[device_ID]==1)
            {
                //Turn OFF
                SWITCH_COMMAND(0,device_ID,0);
            }
        }
        //Change switch status
        lamp_switch[device_ID]=0;
    }
}
else
{
    //If its switch is released
    if (input(PIN_B1)==1)
    {

```

```

//Change switch status
lamp_switch[device_ID]=1;
}

//*****AIRCOND 1*****
//If the switch is pressed
device_ID=0;
    if (input(PIN_B2)==0)
{
    //If the switch has been released previously
    if (aircond_switch[device_ID]==1)
    {
        //If the device is currently OFF
        if (aircond[device_ID]==0)
        {
            //Turn ON
            SWITCH_COMMAND(1,device_ID,1);
        }
        else
            //If the device is currently ON
            if (aircond[device_ID]==1)
            {
                //Turn OFF
                SWITCH_COMMAND(1,device_ID,0);
            }
    }

    //Change switch status
    aircond_switch[device_ID]=0;
}
}
else
    //If its switch is released
    if (input(PIN_B2)==1)
{
    //Change switch status
    aircond_switch[device_ID]=1;
}

//*****AIRCOND 2*****
//If the switch is pressed
device_ID=1;
    if (input(PIN_B3)==0)
{
    //If the switch has been released previously
    if (aircond_switch[device_ID]==1)
    {
        //If the device is currently OFF
        if (aircond[device_ID]==0)
        {
            //Turn ON
            SWITCH_COMMAND(1,device_ID,1);
        }
        else
            //If the device is currently ON
            if (aircond[device_ID]==1)
            {
                //Turn OFF
                SWITCH_COMMAND(1,device_ID,0);
            }
    }

    //Change switch status
    aircond_switch[device_ID]=0;
}
}
else
    //If its switch is released
    if (input(PIN_B3)==1)
{
    //Change switch status
    aircond_switch[device_ID]=1;
}

```

```

}

    //*****DOOR 1*****
    //If the sensor is pressed
device_ID=0;
    if (input(PIN_B4)==0)
{
    //If the device is currently Opened
    if (door[device_ID]==1)
    {
        //Turn Closed
        SWITCH_COMMAND(2,device_ID,0);
    }
}
else
    //If its switch is released
    if (input(PIN_B4)==1)
{
    //If the device is currently Closed
    if (door[device_ID]==0)
    {
        //Turn Opened
        SWITCH_COMMAND(2,device_ID,1);
    }
}

    //*****DOOR 2*****
    //If the sensor is pressed
device_ID=1;
    if (input(PIN_B5)==0)
{
    //If the device is currently Opened
    if (door[device_ID]==1)
    {
        //Turn Closed
        SWITCH_COMMAND(2,device_ID,0);
    }
}
else
    //If its switch is released
    if (input(PIN_B5)==1)
{
    //If the device is currently Closed
    if (door[device_ID]==0)
    {
        //Turn Opened
        SWITCH_COMMAND(2,device_ID,1);
    }
}

    //*****ROOMDOOR 1*****
    //If the sensor is pressed
device_ID=0;
    if (input(PIN_B6)==0)
{
    //If the device is currently Opened
    if (roomdoor[device_ID]==1)
    {
        //Turn Closed
        SWITCH_COMMAND(3,device_ID,0);
    }
}
else
    //If its switch is released
    if (input(PIN_B6)==1)
{
    //If the device is currently Closed
    if (roomdoor[device_ID]==0)
    {

```

```

    //Turn Opened
    SWITCH_COMMAND(3,device_ID,1);
}
}

    //*****ROOMDOOR 2*****
    //If the sensor is pressed
device_ID=1;
    if (input(PIN_B7)==0)
{
    //If the device is currently Opened
    if (roomdoor[device_ID]==1)
    {
        //Turn Closed
        SWITCH_COMMAND(3,device_ID,0);
    }
}
else
    //If its switch is released
    if (input(PIN_B7)==1)
{
    //If the device is currently Closed
    if (roomdoor[device_ID]==0)
    {
        //Turn Opened
        SWITCH_COMMAND(3,device_ID,1);
    }
}
}
}

void REPLY_INITIALIZATION(void)
{
    int reply_array[4];
    int count1;
    int count2;

    //Send all status of device
    //For lamp n aircond only the one that currently ON
    //For door n roomdoor only the one that currently Opened
    for (count2=0;count2<=device_qty-1;count2++)
    {
        //Reply for lamp
        if (lamp[count2]==1)
        {
            //Set reply
            reply_array[0]=0x03;
            reply_array[1]=0x00;
            reply_array[2]=count2;
            reply_array[3]=0x01;

            //Send reply
            for (count1=0;count1<=3;count1++)
            {
                putchar(reply_array[count1]);
            }

            delay_ms(100);
        }

        //Reply for aircond
        if (aircond[count2]==1)
        {
            //Set reply
            reply_array[0]=0x03;
            reply_array[1]=0x01;
            reply_array[2]=count2;
        }
    }
}

```



```

        reply_array[3]=0x01;

        //Send reply
        for (count1=0;count1<=3;count1++)
        {
            putchar(reply_array[count1]);
        }

        delay_ms(100);
    }

//Reply for door
    if (door[count2]==1)
    {
        //Set reply
        reply_array[0]=0x03;
        reply_array[1]=0x02;
        reply_array[2]=count2;
        reply_array[3]=0x01;

        //Send reply
        for (count1=0;count1<=3;count1++)
        {
            putchar(reply_array[count1]);
        }

        delay_ms(100);
    }

//Reply for roomdoor
    if (roomdoor[count2]==1)
    {
        //Set reply
        reply_array[0]=0x03;
        reply_array[1]=0x03;
        reply_array[2]=count2;
        reply_array[3]=0x01;

        //Send reply
        for (count1=0;count1<=3;count1++)
        {
            putchar(reply_array[count1]);
        }

        delay_ms(100);
    }
}

//Set reply
reply_array[0]=0x00;
reply_array[1]=0x00;
reply_array[2]=0x00;
reply_array[3]=0x00;

//Send reply
for (count1=0;count1<=3;count1++)
{
    putchar(reply_array[count1]);
}
}

```

Figure A-15: PIC C Language Program

Visual Basic Coding

```

'Utk form resize property
Public tinggi_form_lama
Public lebar_form_lama
Public tinggi_form_baru
Public lebar_form_baru
Public constraint_tinggi
Public constraint_lebar

'Problem note
'masalah nak initialize door n roomdoor, kena ask controller on start
'array door n roomdoor asalnya 0=closed (tp gui tunjuk 1=opened)
'masalah nak initialize lamp n aircond, kena ask controller on start

'Progress Note

'controller
'-----
'CTRL_SERIAL_CONNECT=complete
'CTRL_SERIAL_DISCONNECT=complete
'tgl_ctrlconnect_click=complete
'ENABLE_CTRL_OBJECT=complete

'STATUS_CTRLTX_INITIALIZATION=complete
'STATUS_CTRLRX_INITIALIZATION=complete
'STATUS_CTRLTX_REINITIALIZATION=complete
'STATUS_CTRLRX_REINITIALIZATION=complete

'MSCOMM_ctrl_oncomm= in progress

'fbus
'-----
'FBUS_SERIAL_CONNECT=complete
'FBUS_SERIAL_DISCONNECT=complete
'tgl_fbuseconnect_click=complete
'ENABLE_FBUS_OBJECT=complete

'STATUS_FBUSTX_REINITIALIZATION=complete
'STATUS_FBUSRX_REINITIALIZATION=complete

'BASE_CONVERTER=complete
'form_unload=complete
'form_resize=complete

'switches
'-----
'tgl_lamp_Click=complete
'tgl_aircond_Click=complete

'send data
'-----
'WRITE_TERMINAL_STATUS=complete <- one of the component to write status in send data
'SWITCH_COMMAND=complete <- reflex the ctrl tx

'Utk connection
Public ctrl_currentport
Public ctrl_currentbaudrate
Public ctrl_is_connected

Public fbus_currentport
Public fbus_is_connected
Public fbus_start_detection

'Utk data counter
Public ctrl_tx_counter

```

```

Public ctrl_rx_counter

Public fbus_tx_counter
Public fbus_rx_counter

'Utk flex
Public ctrl_row_rx_counter
Public ctrl_row_tx_counter

'Utk array device status
'device_status_array(device_type,device_no)
Dim device_status_array(4, 2)

'Utk atasi masalah togglebutton
Public serialcommand_dir

Private Sub Form_Load()

'Constant utk form resize property
tinggi_form_lama = Height
tinggi_form_baru = 0
lebar_form_lama = Width
lebar_form_baru = 0
constraint_tinggi = 7245
constraint_lebar = 9255

'All initialization
Call FORM_INITIALIZATION

Call STATUS_CTRLTX_INITIALIZATION
Call STATUS_CTRLRX_INITIALIZATION

Call ENABLE_CTRL_OBJECT(False)
Call ENABLE_FBUS_OBJECT(False)

'Load Form2
Form2.Show 0, Me

'Write status
textmessage = Me.Caption & " started"
WRITE_STATUS (textmessage)
textmessage = "Controller connection has not yet been established"
WRITE_STATUS (textmessage)
textmessage = "FBUS connection has not yet been established"
WRITE_STATUS (textmessage)

'testaje
'Call WRITE_TERMINAL_STATUS(1, "testaje")
'Call WRITE_TERMINAL_STATUS(1, "testaje")
' Call WRITE_TERMINAL_STATUS(1, "testaje")
' Call WRITE_TERMINAL_STATUS(1, "testaje")
' Call WRITE_TERMINAL_STATUS(2, "testaje")
' Call WRITE_TERMINAL_STATUS(2, "testaje")
' Call WRITE_TERMINAL_STATUS(3, "testaje")
' Call WRITE_TERMINAL_STATUS(3, "testaje")
' Call WRITE_TERMINAL_STATUS(4, "testaje")
' Call WRITE_TERMINAL_STATUS(4, "testaje")
'Call SWITCH_COMMAND(1, 1, 1)
'Call SWITCH_COMMAND(1, 0, 1)

'tgl_ctrlconnect.Value = True
End Sub

'COMPLETE-CHECKED
Private Sub Form_Unload(Cancel As Integer)
Call CTRL_SERIAL_DISCONNECT
Call FBUS_SERIAL_DISCONNECT
End Sub

Private Sub FORM_INITIALIZATION()

```

```

'Initialization
Me.Caption = App.Title & " v" & App.Major & "." & App.Minor & App.Revision

'Combo box initialization
'Put value in the combo box
array_ctrlport = Array("1", "2")
array_fbusport = Array("1", "2")

For count1 = 0 To 1
    cmb_ctrlport.List(count1) = array_ctrlport(count1)
Next
For count1 = 0 To 1
    cmb_fbusport.List(count1) = array_fbusport(count1)
Next

'Set initial value
cmb_ctrlport.ListIndex = 0
cmb_fbusport.ListIndex = 1

'Set initial global value
ctrl_currentport = cmb_ctrlport.List(cmb_ctrlport.ListIndex)
ctrl_currentbaudrate = "9600"

fbus_currentport = cmb_fbusport.List(cmb_fbusport.ListIndex)

ctrl_is_connected = False
fbus_is_connected = False
fbus_start_detection = False

ctrl_tx_counter = 0
ctrl_rx_counter = 0
fbus_tx_counter = 0
fbus_rx_counter = 0

'Set initial device status
'0=OFF,1=ON
'0=closed,1=opened
For count1 = 0 To 3
    For count2 = 0 To 1
        device_status_array(count1, count2) = 0
    Next
Next

'serialcommand_dir=0 (dtg dr komp), serialcommand_dir=1 (dtg dr controller)
serialcommand_dir = 0
End Sub

```

```

'COMPLETE-CHECKED
Private Sub STATUS_CTRLTX_INITIALIZATION()

```

```

    ctrl_row_tx_counter = 0

    'Column Width
    Form2.flex_ctrlstatus_tx.ColWidth(0, 0) = 500
    Form2.flex_ctrlstatus_tx.ColWidth(1, 0) = 400
    Form2.flex_ctrlstatus_tx.ColWidth(2, 0) = 400
    Form2.flex_ctrlstatus_tx.ColWidth(3, 0) = 400
    Form2.flex_ctrlstatus_tx.ColWidth(4, 0) = 820
    Form2.flex_ctrlstatus_tx.ColWidth(5, 0) = 500

    'Write title
    Form2.flex_ctrlstatus_tx.TextMatrix(0, 0) = "No"
    Form2.flex_ctrlstatus_tx.TextMatrix(0, 1) = "Set"
    Form2.flex_ctrlstatus_tx.TextMatrix(0, 2) = "Byte"
    Form2.flex_ctrlstatus_tx.TextMatrix(0, 3) = "Dec"
    Form2.flex_ctrlstatus_tx.TextMatrix(0, 4) = "Bin"
    Form2.flex_ctrlstatus_tx.TextMatrix(0, 5) = "Hex"

    'Alignment

```

```

Form2.flex_ctrlstatus_tx.ColAlignmentFixed(0) = 4
Form2.flex_ctrlstatus_tx.ColAlignmentFixed(1) = 4
Form2.flex_ctrlstatus_tx.ColAlignmentFixed(2) = 4
Form2.flex_ctrlstatus_tx.ColAlignmentFixed(3) = 4
Form2.flex_ctrlstatus_tx.ColAlignmentFixed(4) = 4
Form2.flex_ctrlstatus_tx.ColAlignmentFixed(5) = 4

```

```
Form2.flex_ctrlstatus_tx.ColAlignment(2) = 1
```

```
End Sub
```

```
'COMPLETE-CHECKED
```

```
Private Sub STATUS_CTRLRX_INITIALIZATION()
```

```
ctrl_row_rx_counter = 0
```

```
'Column Width
```

```
Form2.flex_ctrlstatus_rx.ColWidth(0, 0) = 500
Form2.flex_ctrlstatus_rx.ColWidth(1, 0) = 400
Form2.flex_ctrlstatus_rx.ColWidth(2, 0) = 400
Form2.flex_ctrlstatus_rx.ColWidth(3, 0) = 400
Form2.flex_ctrlstatus_rx.ColWidth(4, 0) = 820
Form2.flex_ctrlstatus_rx.ColWidth(5, 0) = 500
```

```
'Write title
```

```
Form2.flex_ctrlstatus_rx.TextMatrix(0, 0) = "No"
Form2.flex_ctrlstatus_rx.TextMatrix(0, 1) = "Set"
Form2.flex_ctrlstatus_rx.TextMatrix(0, 2) = "Byte"
Form2.flex_ctrlstatus_rx.TextMatrix(0, 3) = "Dec"
Form2.flex_ctrlstatus_rx.TextMatrix(0, 4) = "Bin"
Form2.flex_ctrlstatus_rx.TextMatrix(0, 5) = "Hex"
```

```
'Alignment
```

```
Form2.flex_ctrlstatus_rx.ColAlignmentFixed(0) = 4
Form2.flex_ctrlstatus_rx.ColAlignmentFixed(1) = 4
Form2.flex_ctrlstatus_rx.ColAlignmentFixed(2) = 4
Form2.flex_ctrlstatus_rx.ColAlignmentFixed(3) = 4
Form2.flex_ctrlstatus_rx.ColAlignmentFixed(4) = 4
Form2.flex_ctrlstatus_rx.ColAlignmentFixed(5) = 4
```

```
Form2.flex_ctrlstatus_rx.ColAlignment(2) = 1
```

```
End Sub
```

```
'COMPLETE-CHECKED
```

```
Public Sub STATUS_CTRLTX_REINITIALIZATION()
```

```
ctrl_row_tx_counter = 0
```

```
Form2.txt_ctrlstatus_tx.TextRTF = ""
```

```
Form2.flex_ctrlstatus_tx.Rows = 2
Form2.flex_ctrlstatus_tx.TextMatrix(1, 1) = ""
Form2.flex_ctrlstatus_tx.TextMatrix(1, 2) = ""
Form2.flex_ctrlstatus_tx.TextMatrix(1, 3) = ""
Form2.flex_ctrlstatus_tx.TextMatrix(1, 4) = ""
Form2.flex_ctrlstatus_tx.TextMatrix(1, 5) = ""
```

```
End Sub
```

```
'COMPLETE-CHECKED
```

```
Public Sub STATUS_CTRLRX_REINITIALIZATION()
```

```
ctrl_row_rx_counter = 0
```

```
Form2.txt_ctrlstatus_rx.TextRTF = ""
```

```
Form2.flex_ctrlstatus_rx.Rows = 2
Form2.flex_ctrlstatus_rx.TextMatrix(1, 1) = ""
Form2.flex_ctrlstatus_rx.TextMatrix(1, 2) = ""
```

```

Form2.flex_ctrlstatus_rx.TextMatrix(1, 3) = ""
Form2.flex_ctrlstatus_rx.TextMatrix(1, 4) = ""
Form2.flex_ctrlstatus_rx.TextMatrix(1, 5) = ""

End Sub

'COMPLETE-CHECKED
Public Sub STATUS_FBUSTX_REINITIALIZATION()

    Form2.txt_fbusstatus_tx.TextRTF = ""

End Sub

'COMPLETE-CHECKED
Public Sub STATUS_FBUSRX_REINITIALIZATION()

    Form2.txt_fbusstatus_rx.TextRTF = ""

End Sub

'COMPLETE-CHECKED
Private Sub cmd_close_Click()
    Unload Me
End Sub

'COMPLETE-CHECKED
Private Sub cmd_showdata_Click()
    Form2.Show 0, Me
End Sub

'COMPLETE-CHECKED
Private Sub cmd_showascii_Click()
    Form3.Show 0, Me
End Sub

'COMPLETE-CHECKED
Private Sub cmd_showserial_Click()
    Form4.Show 0, Me
End Sub

'COMPLETE-CHECKED
Private Sub cmb_ctrlport_Click()
    ctrl_currentport = cmb_ctrlport.List(cmb_ctrlport.ListIndex)
End Sub

'COMPLETE-CHECKED
Private Sub cmb_fbusport_Click()
    fbus_currentport = cmb_fbusport.List(cmb_fbusport.ListIndex)
End Sub

'COMPLETE-CHECKED
Private Sub WRITE_STATUS(textmessage)
    msg_datetime = Format(Now, "dd/mm/yyyy hh:mm:ss")
    txt_status.TextRTF = "{" & txt_status.TextRTF & "\fi-2160\i2160\b " & msg_datetime & "\b0\tab " & textmessage & "\par}"
    txt_status.TextRTF = "{" & txt_status.TextRTF & "\n}"
    txt_status.SelStart = Len(txt_status.TextRTF)
End Sub

'COMPLETE-CHECKED
Private Sub ENABLE_CTRL_OBJECT(status_object As Boolean)
    If status_object = True Then
        For count1 = 0 To 1
            tgl_lamp(count1).Enabled = True
            tgl_aircond(count1).Enabled = True
            txt_door(count1).Enabled = True
            txt_roomdoor(count1).Enabled = True

            led_lamp(count1).FillColor = &HFF&
            led_aircond(count1).FillColor = &HFF&
            led_door(count1).FillColor = &HFF&
        Next count1
    End If
End Sub

```

```

    led_roomdoor(count1).FillColor = &HFF&
Next

cmb_ctrlport.Enabled = False
ElseIf status_object = False Then
    serialcommand_dir = 1

For count1 = 0 To i
    tgl_lamp(count1).Value = False
    tgl_aircond(count1).Value = False

    tgl_lamp(count1).Caption = "Turn ON"
    tgl_aircond(count1).Caption = "Turn ON"

    tgl_lamp(count1).Enabled = False
    tgl_aircond(count1).Enabled = False

    txt_door(count1).Enabled = False
    txt_roomdoor(count1).Enabled = False

    txt_door(count1).Text = "Closed"
    txt_roomdoor(count1).Text = "Closed"

    led_lamp(count1).FillColor = &HFF8080
    led_aircond(count1).FillColor = &HFF8080
    led_door(count1).FillColor = &HFF8080
    led_roomdoor(count1).FillColor = &HFF8080
Next

serialcommand_dir = 0

cmb_ctrlport.Enabled = True
End If
End Sub

'COMPLETE-CHECKED
Private Sub ENABLE_FBUS_OBJECT(status_object As Boolean)
    If status_object = True Then
        txt_provname.Enabled = True
        txt_provmsc.Enabled = True
        txt_provcountry.Enabled = True
        txt_provcode.Enabled = True
        txt_hwdate.Enabled = True
        txt_hwtime.Enabled = True

        cmb_fbuserport.Enabled = False

    ElseIf status_object = False Then
        txt_provname.Enabled = False
        txt_provmsc.Enabled = False
        txt_provcountry.Enabled = False
        txt_provcode.Enabled = False
        txt_hwdate.Enabled = False
        txt_hwtime.Enabled = False

        cmb_fbuserport.Enabled = True

    End If
End Sub

'COMPLETE-CHECKED
Private Sub cmd_clrstatus_Click()
    txt_status.TextRTF = ""
End Sub

'COMPLETE-CHECKED
Private Sub tgl_lamp_Click(Index As Integer)
    If serialcommand_dir = 0 Then
        If tgl_lamp(Index) = True Then

```

```

    tgl_lamp(Index).Caption = "Turn OFF"
    Call SWITCH_COMMAND(0, Index, 1)
ElseIf tgl_lamp(Index) = False Then
    tgl_lamp(Index).Caption = "Turn ON"
    Call SWITCH_COMMAND(0, Index, 0)
End If
End If
End Sub

'COMPLETE-CHECKED
Private Sub tgl_aircond_Click(Index As Integer)

    If serialcommand_dir = 0 Then
        If tgl_aircond(Index) = True Then
            tgl_aircond(Index).Caption = "Turn OFF"
            Call SWITCH_COMMAND(1, Index, 1)
        ElseIf tgl_aircond(Index) = False Then
            tgl_aircond(Index).Caption = "Turn ON"
            Call SWITCH_COMMAND(1, Index, 0)
        End If
    End If
End Sub

'COMPLETE-CHECKED
Private Sub tgl_ctrlconnect_Click()

    'Connect
    If tgl_ctrlconnect.Value = True Then

        tgl_ctrlconnect.Caption = "Disconnect"
        Call CTRL_SERIAL_CONNECT

        'If succesful
        If ctrl_is_connected = True Then

            'Reset counter
            ctrl_tx_counter = 0
            ctrl_rx_counter = 0
            'Enable object
            Call ENABLE_CTRL_OBJECT(True)

            'Reset flex
            Call STATUS_CTRLTX_REINITIALIZATION
            Call STATUS_CTRLRX_REINITIALIZATION

            'Acquire device status
            Call CONTROLLER_ACQUIRE_INFO

            'If still not connected
            ElseIf ctrl_is_connected = False Then

                'Disable object
                Call ENABLE_CTRL_OBJECT(False)
                tgl_ctrlconnect.Caption = "Connect"
                'Turn OFF button back
                tgl_ctrlconnect.Value = False

            End If

        'Disconnect
        ElseIf tgl_ctrlconnect.Value = False Then

            'Reset initial device status
            '0=OFF,1=ON
            '0=closed,1=opened
            For count1 = 0 To 3
                For count2 = 0 To 1
                    device_status_array(count1, count2) = 0
                Next
            Next
        Next
    End Sub

```



```

'Disable object
Call ENABLE_CTRL_OBJECT(False)
tgl_ctrlconnect.Caption = "Connect"
Call CTRL_SERIAL_DISCONNECT

End If

End Sub

'Belum siap
Private Sub MSComm_ctrl_OnComm()

    Dim rcv_string As String

    Select Case MSComm_ctrl.CommEvent
        Case comEvReceive
            'On receive data...
            rcv_string = MSComm_ctrl.Input

            'Process rcv data
            Call SWITCH_REPLY(rcv_string)
        End Select

    End Sub

'COMPLETE-CHECKED
Private Sub CTRL_SERIAL_CONNECT()
    'set the active serial port
    MSComm_ctrl.CommPort = ctrl_currentport

    'set the baudrate,parity,databits,stopbits for the connection
    MSComm_ctrl.Settings = ctrl_currentbaudrate & ",N,8,1"

    'set the DTR and RTS flags
    MSComm_ctrl.DTREnable = False
    MSComm_ctrl.RTSEnable = False

    'enable the oncomm event for every received character
    'RThreshold=1,comEvReceive=enabled
    'RThreshold=0,comEvReceive=disabled
    MSComm_ctrl.RThreshold = 1

    'disable the oncomm event for send characters
    'SThreshold=1,comEvSend=enabled
    'SThreshold=0,comEvSend=disabled
    MSComm_ctrl.SThreshold = 0

    'Write status
    textmessage = "Connecting to controller..."
    WRITE_STATUS (textmessage)

    On Error GoTo errorhandler
    'open the serial port
    MSComm_ctrl.PortOpen = True
    ctrl_is_connected = True

    'Write status
    textmessage = "Serial connection to the controller has been established successfully on Port " & ctrl_currentport & " in " &
ctrl_currentbaudrate & " baudrate"
    WRITE_STATUS (textmessage)

    'This exit sub is to prevent the normal flow (without error) goes into errorhandler
    Exit Sub

errorhandler:
    A1 = MsgBox(Err.Description & vbCrLf & "[Error no. = " & Err.Number & "]", vbExclamation, "Error")
    ctrl_is_connected = False

    'Write status

```

```
textmessage = Err.Description & ". Serial connection attempt to the controller failed"
WRITE_STATUS (textmessage)
```

```
End Sub
```

```
'COMPLETE-CHECKED
```

```
Private Sub CTRL_SERIAL_DISCONNECT()
```

```
'Close port if and only if it is currently connected
If ctrl_is_connected = True Then
```

```
'Write status
textmessage = "Disconnecting from controller..."
WRITE_STATUS (textmessage)
```

```
MSComm_ctrl.PortOpen = False
ctrl_is_connected = False
```

```
'Write status
textmessage = "Serial connection to the controller has been closed"
WRITE_STATUS (textmessage)
```

```
End If
End Sub
```

```
'COMPLETE-CHECKED
```

```
Private Sub tgl_fbusconnect_Click()
```

```
'Connect
If tgl_fbusconnect.Value = True Then
```

```
tgl_fbusconnect.Caption = "Disconnect"
Call FBUS_SERIAL_CONNECT
```

```
'If succesful
If fbus_is_connected = True Then
```

```
'Reset counter
fbus_tx_counter = 0
fbus_rx_counter = 0
'Enable object
Call ENABLE_FBUS_OBJECT(True)
'Acquire info
Call FBUS_ACQUIRE_INFO(0)
'Delete old msg
Call SERVER_INITIALIZATION
```

```
'Reset flex
Call STATUS_FBUSTX_REINITIALIZATION
Call STATUS_FBUSRX_REINITIALIZATION
```

```
'Start detection
fbus_start_detection = True
```

```
'If still not connected
ElseIf fbus_is_connected = False Then
```

```
'Disable object
Call ENABLE_FBUS_OBJECT(False)
tgl_fbusconnect.Caption = "Connect"
'Turn OFF button back
tgl_fbusconnect.Value = False
```

```
End If
```

```
'Disconnect
```

```
ElseIf tgl_fbusconnect.Value = False Then
```

```
'Disable object
Call ENABLE_FBUS_OBJECT(False)
```

```

    tgl_fbconnect.Caption = "Connect"
    Call FBUS_SERIAL_DISCONNECT

End If

End Sub

'COMPLETE-CHECKED
Private Sub FBUS_SERIAL_CONNECT()

    'Write status
    textmessage = "Connecting to server's FBUS..."
    WRITE_STATUS (textmessage)
    DoEvents

    On Error GoTo errorhandler
    'open the serial port
    MFBUS15Control1.Connect "COM" & fbus_currentport
    fbus_is_connected = True

    'Write status
    textmessage = "Serial connection to the server's FBUS has been established successfully on Port " & fbus_currentport & " in
    115200 baudrate"
    WRITE_STATUS (textmessage)

    'This exit sub is to prevent the normal flow (without error) goes into errorhandler
    Exit Sub

errorhandler:
    A1 = MsgBox(Err.Description, vbExclamation, "Error")
    fbus_is_connected = False

    'Write status
    textmessage = Err.Description & ". Serial connection attempt to the server's FBUS failed"
    WRITE_STATUS (textmessage)

End Sub

'COMPLETE-CHECKED
Private Sub FBUS_SERIAL_DISCONNECT()

    'Close port if and only if it is currently connected
    If fbus_is_connected = True Then

        'Write status
        textmessage = "Disconnecting from server's FBUS..."
        WRITE_STATUS (textmessage)
        DoEvents

        MFBUS15Control1.Disconnect
        fbus_is_connected = False

        'Write status
        textmessage = "Serial connection to the server's FBUS has been closed"
        WRITE_STATUS (textmessage)

    End If

End Sub

txt_provname.Text = MFBUS15Control1.ProviderName
txt_provmsc.Text = MFBUS15Control1.ProviderSMSC
txt_provcountry.Text = MFBUS15Control1.ProviderCountry
txt_provcode.Text = MFBUS15Control1.ProviderCode
txt_hwdate.Text = Format(MFBUS15Control1.DateTime, "DD/MM/YYYY")
txt_hvertime.Text = Format(MFBUS15Control1.DateTime, "hh:mm:ss")

'Write status
textmessage = "Information from server's FBUS successfully retrieved..."
WRITE_STATUS (textmessage)

```

```

ElseIf data_type = 1 Then
    txt_hwdate.Text = Format(MFBUS15Control1.DateTime, "DD/MM/YYYY")
    txt_hwtime.Text = Format(MFBUS15Control1.DateTime, "hh:mm:ss")

End If
End If

End Sub

'COMPLETE-CHECKED
Private Sub CONTROLLER_ACQUIRE_INFO()

    ctrl_tx_array = Array(&HAA, &H0, &H0, &H0, &H0, &H0, &H0, &H0, &HFF)

    'Append all bytes again
    command_message = ""
    For count1 = 0 To 8
        command_message = command_message & Chr(ctrl_tx_array(count1))
    Next

    'Write status
    textmessage = "Retrieving initial device status from controller..."
    WRITE_STATUS(textmessage)

    'Send via serial to controller!
    MSComm_ctrl.Output = command_message

    'Increase transmit counter
    ctrl_tx_counter = ctrl_tx_counter + 1
    'Write status and flexgrid
    Call WRITE_TERMINAL_STATUS(1, command_message)

End Sub

'COMPLETE-CHECKED
Private Sub Form_Resize()
    'CONSTANT
    lebar_form = Width
    tinggi_form = Height

    'Set minimum constraint for height
    If tinggi_form >= constraint_tinggi Then
        'Renew value
        tinggi_form_baru = tinggi_form
    ElseIf tinggi_form < constraint_tinggi Then
        'Renew value
        tinggi_form_baru = constraint_tinggi
    End If

    bezatinggi_form = tinggi_form_baru - tinggi_form_lama
    tinggi_form_lama = tinggi_form_baru

    'Reset object
    cmd_close.Top = cmd_close.Top + bezatinggi_form
    cmd_clrstatus.Top = cmd_clrstatus.Top + bezatinggi_form
    cmd_showdata.Top = cmd_showdata.Top + bezatinggi_form
    cmd_showascii.Top = cmd_showascii.Top + bezatinggi_form
    cmd_showserial.Top = cmd_showserial.Top + bezatinggi_form

    txt_status.Height = txt_status.Height + bezatinggi_form
    frm_status.Height = frm_status.Height + bezatinggi_form

    'Set minimum constraint for width
    If lebar_form >= constraint_lebar Then
        'Renew value
        lebar_form_baru = lebar_form
    ElseIf lebar_form < constraint_lebar Then
        'Renew value
        lebar_form_baru = constraint_lebar
    End If

```

```
bezalebar_form = lebar_form_baru - lebar_form_lama
lebar_form_lama = lebar_form_baru
```

```
'Reset object
cmd_close.Left = cmd_close.Left + bezalebar_form
cmd_clrstatus.Left = cmd_clrstatus.Left + bezalebar_form
cmd_showdata.Left = cmd_showdata.Left + bezalebar_form
cmd_showascii.Left = cmd_showascii.Left + bezalebar_form
cmd_showserial.Left = cmd_showserial.Left + bezalebar_form
```

```
txt_status.Width = txt_status.Width + bezalebar_form
frm_status.Width = frm_status.Width + bezalebar_form
```

```
txt_provname.Width = txt_provname.Width + bezalebar_form
txt_provmsc.Width = txt_provmsc.Width + bezalebar_form
txt_provcountry.Width = txt_provcountry.Width + bezalebar_form
txt_provcode.Width = txt_provcode.Width + bezalebar_form
txt_hwdate.Width = txt_hwdate.Width + bezalebar_form
txt_hwtime.Width = txt_hwtime.Width + bezalebar_form
```

```
frm_info.Width = frm_info.Width + bezalebar_form
```

End Sub

'COMPLETE-CHECKED

Private Function BASE_CONVERTER(input_number, mode_from, mode_to)

```
'0=Binary
'1=Decimal
'2=Hexadecimal
```

```
'Just for hex
array_hex = Array("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F")
```

Convert from input base to decimal

```
'dec->dec
If mode_from = 1 Then
```

```
    inputdecimal = input_number
```

```
'bin->dec
Elseif mode_from = 0 Then
```

```
    bufferanswer = 0
```

```
    For count1 = 1 To Len(input_number)
        accumulate = (Mid(input_number, Len(input_number) - count1 + 1, 1)) * (2 ^ (count1 - 1))
```

```
        bufferanswer = bufferanswer + accumulate
    Next
```

```
    inputdecimal = bufferanswer
```

```
'hex->dec
Elseif mode_from = 2 Then
```

```
    bufferanswer = 0
```

```
    For count1 = 1 To Len(input_number)
```

```
        For count2 = 0 To 15
            If array_hex(count2) = (UCASE(Mid(input_number, Len(input_number) - count1 + 1, 1))) Then
                chartoint = count2
            Exit For
        End If
    Next
```

```
        accumulate = chartoint * (16 ^ (count1 - 1))
```

```

    bufferanswer = bufferanswer + accumulate
Next

inputdecimal = bufferanswer

End If
***End conversion***

***Convert from decimal to required base***
'dec->dec
If mode_to = 1 Then

    returnvalue = inputdecimal

'dec->bin
ElseIf mode_to = 0 Then

    bufferanswer = ""
    bufferinput = inputdecimal

    Do
        remainder = ((bufferinput / 2) - Int(bufferinput / 2)) * 2

        bufferanswer = remainder & bufferanswer

        bufferinput = Int(bufferinput / 2)

        If bufferinput = 0 Then Exit Do
    Loop

    returnvalue = bufferanswer

'dec->hex
ElseIf mode_to = 2 Then

    bufferanswer = ""
    bufferinput = inputdecimal

    Do
        remainder = ((bufferinput / 16) - Int(bufferinput / 16)) * 16

        bufferanswer = array_hex(remainder) & bufferanswer

        bufferinput = Int(bufferinput / 16)

        If bufferinput = 0 Then Exit Do
    Loop

    returnvalue = bufferanswer

End If
***End conversion***

BASE_CONVERTER = returnvalue

End Function

'COMPLETE-CHECKED
Private Sub SWITCH_COMMAND(device_type, device_no, onoff_state)

    'Note
    'device_type=0    lamp
    'device_type=1    aircond
    'device_no=0      device number 1
    'device_no=1      device number 2
    'onoff_state=0    turn OFF

```

```

'onoff_state=1    turn ON

'General terminal->controller commands
'Set again for the 2nd,3rd and 4th last bytes
'2nd last byte=ON/OFF
'3rd last byte=device no
'4th last byte=device type
ctrl_tx_array = Array(&HAA, &H3, &H16, &H1, &H4, &H0, &H0, &H0, &HFF)
ctrl_tx_array(5) = device_type
ctrl_tx_array(6) = device_no
ctrl_tx_array(7) = onoff_state

'Append all bytes again
command_message = ""
For count1 = 0 To 8
    command_message = command_message & Chr(ctrl_tx_array(count1))
Next

'Send via serial to controller!
MSComm_ctrl.Output = command_message

'Write status
Select Case device_type
    Case 0
        device_type_desc = "Lamp"
    Case 1
        device_type_desc = "Air-conditioner"
End Select
Select Case device_no
    Case 0
        device_no_desc = "1"
    Case 1
        device_no_desc = "2"
End Select
Select Case onoff_state
    Case 0
        onoff_state_desc = "OFF"
    Case 1
        onoff_state_desc = "ON"
End Select

textmessage = "Switching " & onoff_state_desc & " " & device_type_desc & " " & device_no_desc & "..."
WRITE_STATUS (textmessage)

'Increase transmit counter
ctrl_tx_counter = ctrl_tx_counter + 1
'Write status and flexgrid
Call WRITE_TERMINAL_STATUS(1, command_message)

End Sub

'COMPLETE-CHECKED
Private Sub SWITCH_REPLY(rcv_string)

'Increase receive counter
ctrl_rx_counter = ctrl_rx_counter + 1
'Write status
Call WRITE_TERMINAL_STATUS(2, rcv_string)

cmd_group = Asc(Mid(rcv_string, 1, 1))
device_type = Asc(Mid(rcv_string, 2, 1))
device_no = Asc(Mid(rcv_string, 3, 1))
onoff_status = Asc(Mid(rcv_string, 4, 1))

If cmd_group = 3 Then
    serialcommand_dir = 1
    'Note
    'Lamp & Aircond (0=OFF,1=ON)
    'Door & roomdoor (0=Closed,1=Open)

```

```

'LAMP
If device_type = 0 Then
  If onoff_status = 0 Then
    'Up button
    tgl_lamp(device_no).Value = False
    tgl_lamp(device_no).Caption = "Turn ON"
    'Red
    led_lamp(device_no).FillColor = &HFF&

    'Write status
    textmessage = "Lamp " & (device_no + 1) & " has been turned OFF"
    WRITE_STATUS (textmessage)
  Elseif onoff_status = 1 Then
    'Down button
    tgl_lamp(device_no).Value = True
    tgl_lamp(device_no).Caption = "Turn OFF"
    'Green
    led_lamp(device_no).FillColor = &HFF00&

    'Write status
    textmessage = "Lamp " & (device_no + 1) & " has been turned ON"
    WRITE_STATUS (textmessage)
  End If

'AIRCOND
Elseif device_type = 1 Then
  If onoff_status = 0 Then
    'Up button
    tgl_aircond(device_no).Value = False
    tgl_aircond(device_no).Caption = "Turn ON"
    'Red
    led_aircond(device_no).FillColor = &HFF&

    'Write status
    textmessage = "Air-conditioner " & (device_no + 1) & " has been turned OFF"
    WRITE_STATUS (textmessage)
  Elseif onoff_status = 1 Then
    'Down button
    tgl_aircond(device_no).Value = True
    tgl_aircond(device_no).Caption = "Turn OFF"
    'Green
    led_aircond(device_no).FillColor = &HFF00&

    'Write status
    textmessage = "Air-conditioner " & (device_no + 1) & " has been turned ON"
    WRITE_STATUS (textmessage)
  End If

'DOOR SENSOR
Elseif device_type = 2 Then
  If onoff_status = 0 Then
    txt_door(device_no).Text = "Closed"
    'Red
    led_door(device_no).FillColor = &HFF&

    'Write status
    textmessage = "Door " & (device_no + 1) & " has been closed"
    WRITE_STATUS (textmessage)
  Elseif onoff_status = 1 Then
    txt_door(device_no).Text = "Opened"
    'Green
    led_door(device_no).FillColor = &HFF00&

    'Write status
    textmessage = "Door " & (device_no + 1) & " has been opened"
    WRITE_STATUS (textmessage)
  End If

```



```

ROOM-DOOR SENSOR
Elseif device_type = 3 Then
  If onoff_status = 0 Then
    txt_roomdoor(device_no).Text = "Closed"
    'Red
    led_roomdoor(device_no).FillColor = &HFF&

    'Write status
    textmessage = "Room door " & (device_no + 1) & " has been closed"
    WRITE_STATUS (textmessage)

  Elseif onoff_status = 1 Then
    txt_roomdoor(device_no).Text = "Opened"
    'Green
    led_roomdoor(device_no).FillColor = &HFF00&

    'Write status
    textmessage = "Room door " & (device_no + 1) & " has been opened"
    WRITE_STATUS (textmessage)
  End If

End If

End If
serialcommand_dir = 0

'Change status
device_status_array(device_type, device_no) = onoff_status

'testajc
'textmessage = "device_status_array(" & device_type & ", " & device_no & ")=" & device_status_array(device_type, device_no)
'WRITE_STATUS (textmessage)

Elseif cmd_group = 0 Then

  'Complete initial device status retrieval
  If device_type = 0 And device_no = 0 And onoff_status = 0 Then
    textmessage = "Initial device status has been retrieved"
    WRITE_STATUS (textmessage)
    textmessage = "Controller is ready"
    WRITE_STATUS (textmessage)
  End If

End If

End Sub

'COMPLETE-CHECKED
Private Sub WRITE_TERMINAL_STATUS(flex_id, data_string)

  'Note
  'flex_id=1 controller tx
  'flex_id=2 controller rx
  'flex_id=3 fbus tx
  'flex_id=4 fbus rx

  Dim flex_output As Object
  Dim txt_output As Object

  'Check for flex_id
  Select Case flex_id
    Case 1
      Set flex_output = Form2.flex_ctrlstatus_tx
      Set txt_output = Form2.txt_ctrlstatus_tx
      data_counter = ctrl_tx_counter
      row_counter = ctrl_row_tx_counter

    Case 2
      Set flex_output = Form2.flex_ctrlstatus_rx
      Set txt_output = Form2.txt_ctrlstatus_rx
      data_counter = ctrl_rx_counter
  End Select

```

```

row_counter = ctrl_row_rx_counter

Case 3
Set txt_output = Form2.txt_fbusstatus_tx
data_counter = fbus_tx_counter

Case 4
Set txt_output = Form2.txt_fbusstatus_rx
data_counter = fbus_rx_counter
End Select

If flex_id = 1 Or flex_id = 2 Then
'Start write status
textmessage = "Set " & data_counter & " (" & Len(data_string) & " bytes) = ["

'Run conversion to flexgrid
If Len(data_string) > 0 Then
flex_output.Rows = flex_output.Rows + Len(data_string)

For count1 = 1 To Len(data_string)
'Number
flex_output.TextMatrix(row_counter + count1, 0) = row_counter + count1

'Set
flex_output.TextMatrix(row_counter + count1, 1) = data_counter

'Byte
flex_output.TextMatrix(row_counter + count1, 2) = (Mid(data_string, count1, 1))

'Decimal
flex_output.TextMatrix(row_counter + count1, 3) = Asc(Mid(data_string, count1, 1))

'Binary
buffer1 = BASE_CONVERTER(Asc(Mid(data_string, count1, 1)), 1, 0)
If Len(buffer1) < 8 Then
loopadd = 8 - Len(buffer1)

For count2 = 1 To loopadd
buffer1 = "0" & buffer1
Next
End If
flex_output.TextMatrix(row_counter + count1, 4) = buffer1

'Hex
buffer2 = BASE_CONVERTER(Asc(Mid(data_string, count1, 1)), 1, 2)
If Len(buffer2) < 2 Then
loopadd = 2 - Len(buffer2)

For count2 = 1 To loopadd
buffer2 = "0" & buffer2
Next
End If
flex_output.TextMatrix(row_counter + count1, 5) = "0x" & buffer2

'Continue to write status
textmessage = textmessage & " 0x" & buffer2
Next

'Finally update actual public data
'Check for flex_id
Select Case flex_id
Case 1
ctrl_row_tx_counter = ctrl_row_tx_counter + Len(data_string)
Case 2
ctrl_row_rx_counter = ctrl_row_rx_counter + Len(data_string)
End Select

End If

'Continue to write status

```

```

msg_datetime = Format(Now, "dd/mm/yyyy hh:mm:ss")
txt_output.TextRTF = "{" & txt_output.TextRTF & "\fi-2160\i2160\b " & msg_datetime & "\b0\tab " & textmessage & " ]\par}"
txt_output.TextRTF = "{" & txt_output.TextRTF & "\n}"
txt_output.SelStart = Len(txt_output.TextRTF)

Elseif flex_id = 3 Or flex_id = 4 Then

'Write status
textmessage = "Set " & data_counter & " (" & Len(data_string) & " bytes) = " & data_string & ""

msg_datetime = Format(Now, "dd/mm/yyyy hh:mm:ss")
txt_output.TextRTF = "{" & txt_output.TextRTF & "\fi-2160\i2160\b " & msg_datetime & "\b0\tab " & textmessage & " \par}"
txt_output.TextRTF = "{" & txt_output.TextRTF & "\n}"
txt_output.SelStart = Len(txt_output.TextRTF)

End If

End Sub

Private Sub Timer1_Timer()
'This is date & time updater!
If fbus_is_connected Then
'Acquire time n date only
Call FBUS_ACQUIRE_INFO(1)
End If
End Sub

Private Sub Timer2_Timer()
'This is received msg detector

If fbus_is_connected And fbus_start_detection Then
'Need to refresh first
MFBUS15Control1.SMS.Refresh
'Scan how many msg in inbox
msg_qty = MFBUS15Control1.SMS.Inbox.Count

'Write status
textmessage = "Number of messages=" & msg_qty
WRITE_STATUS (textmessage)

If msg_qty > 0 Then
'Read only the first msg!!!
'Write status
msg_sender_no = MFBUS15Control1.SMS.Inbox.Item(1).Sender
msg_text = MFBUS15Control1.SMS.Inbox.Item(1).Text
'msg_date = Format(MFBUS15Control1.SMS.Inbox.Item(1).DateTime, "DD/MM/YYYY")
'msg_time = Format(MFBUS15Control1.SMS.Inbox.Item(1).DateTime, "hh:mm:ss")

'Increase receive counter
fbus_rx_counter = fbus_rx_counter + 1

'Write status
textmessage = "Command message " & fbus_rx_counter & " has been received.Sender=" & msg_sender_no & ", Message="
& msg_text
WRITE_STATUS (textmessage)

'Write terminal status
command_message = msg_text
Call WRITE_TERMINAL_STATUS(4, command_message)
DoEvents

Call SMS_MESSAGE_PROCESSOR(msg_text, msg_sender_no)

'Delete the first msg (tak kisah berapa byk yg sampai)
'Next msg will be execute next cycle
MFBUS15Control1.SMS.Inbox.Item(1).Delete
End If

```

```
End If
End Sub
Private Sub SMS_MESSAGE_PROCESSOR(sms_string, sender_no)

    If fbus_is_connected And fbus_start_detection Then

        'Convert to uppercase
        sms_string_ucase = UCase(sms_string)

        'Check commands
        Select Case sms_string_ucase
            'System info commands
            Case "I-UTP ACAD22 SYSTEM INFO"

                If ctrl_is_connected = True Then

                    send_msg = "I-UTP ACAD22-01-04 System OK"
                    MFBUS15Control1.SMS.SendMessage sender_no, send_msg

                ElseIf ctrl_is_connected = False Then

                    send_msg = "I-UTP ACAD22-01-04 System not OK. Terminal is disconnected from Controller"
                    MFBUS15Control1.SMS.SendMessage sender_no, send_msg

                End If

            End Case

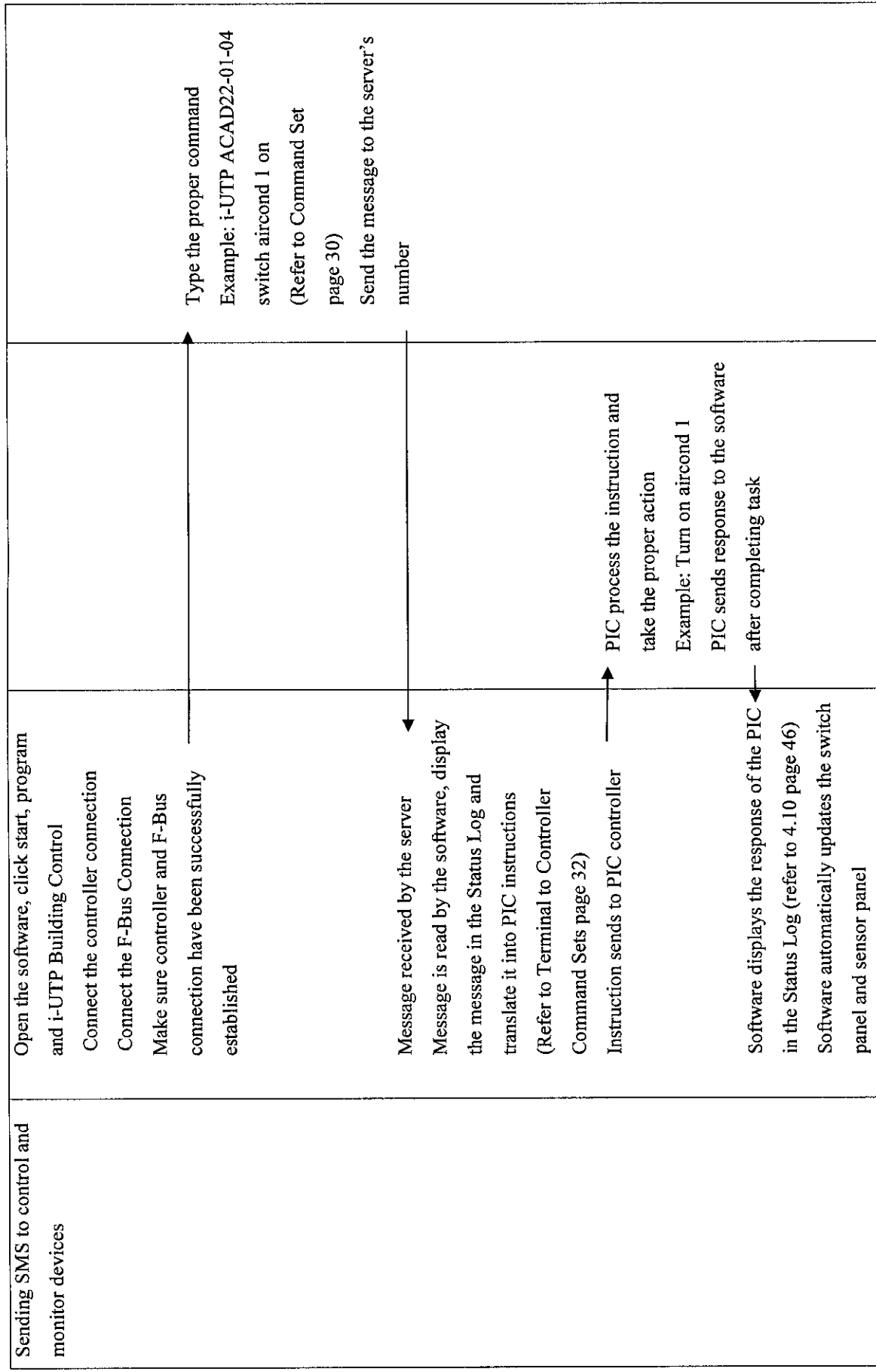
        End Select

    End If

End Sub
```

Figure A-16: Visual Basic Code

PROCESS/PROCEDURE	PC/MONITOR/F-BUS SERVER/VB	PIC CONTROLLER/DEVICES	MOBILE PHONE
<p>Software Installation</p>	<ol style="list-style-type: none"> 1. Copy I-UTP Building Control folder and ActiveX installation folder to desktop 2. Open the ActiveX installation folder 3. Follow the instruction in the Readme text file 4. Open the package folder 5. Run the Setup file 6. Wait until installation finishes 7. To open the software, click Start, program and find i-UTP Building Control 		
<p>Hardware Installation</p>	<ol style="list-style-type: none"> 1. Connect Com Port 1 to the serial port located at the circuit 2. Connect Com Port 2 to the F-Bus Server 3. Open the software, click start, program and i-UTP Building Control 4. Connect the controller connection 5. Connect the F-Bus Connection (Server send Provider Server Information to the software- refer to Figure 4.9 page 46) 	<p>Turn on the power supply to the circuit</p> <p>PIC sends current status of the devices to the software</p>	



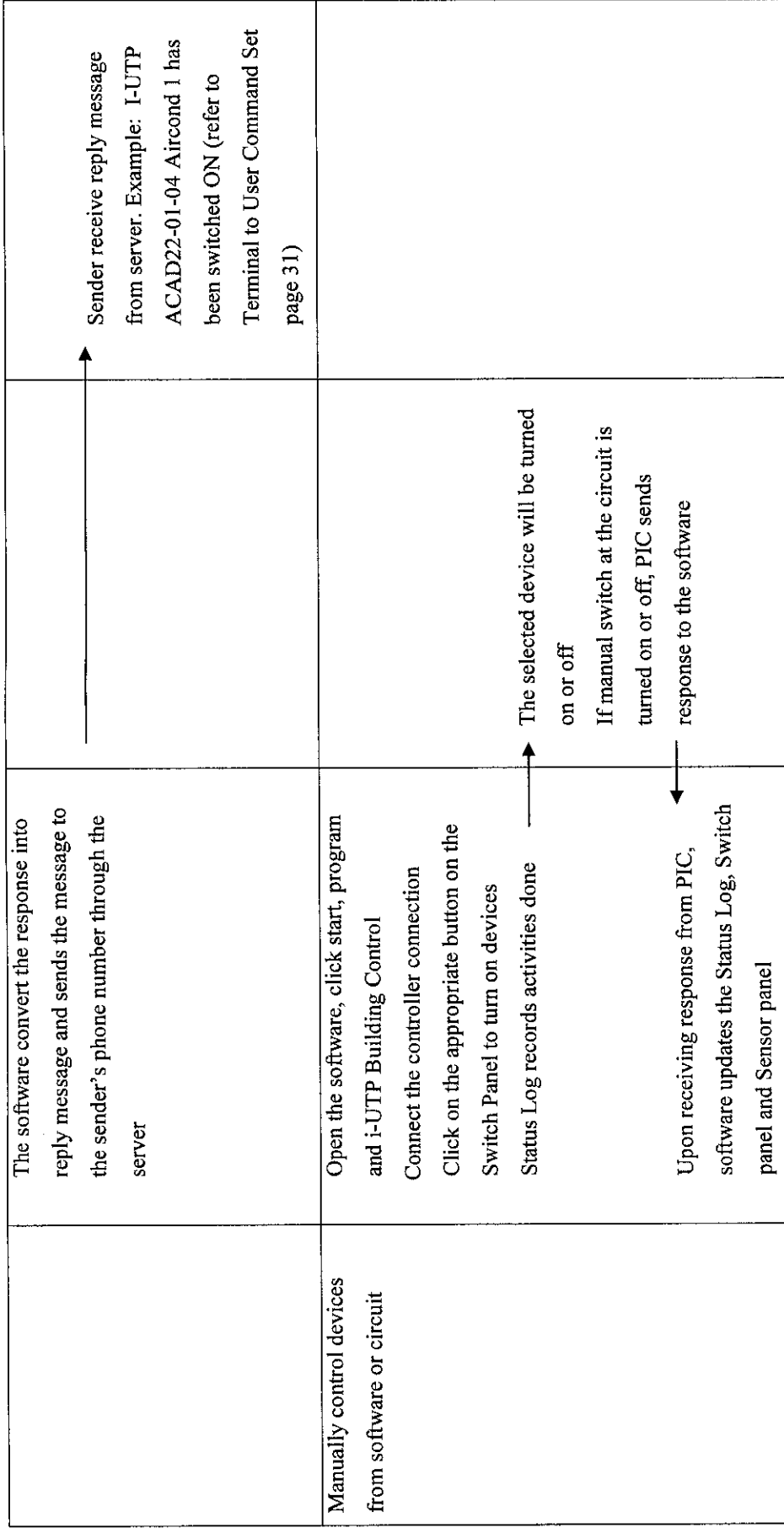


Figure A-17: User Manual

No	Detail/Work	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Selection of Project Topic														
2	Preliminary Research Work														
3	Logbook preparation/ submission														
4	Submission of Preliminary report														
5	Project Work														
	• Reference/ Literature														
	• Research														
	• Practical/ Laboratory Work														
6	Submission of Progress Report														
7	Project Work Continue														
	• Practical/ Laboratory Work														
	• Research														
8	Submission of Interim Report Draft														
9	Submission of Interim Report														

Figure B-1: Semester 1 Gantt chart

No	Detail/Work	1	2	3	4	5	6	7	8	9	10	11	12	13	14	20
1	Research Work															
2	Logbook preparation/ submission															
3	Submission of Progress Report 1															
4	Project Work															
	• Reference/ Literature															
	• Research															
	• Practical/ Laboratory Work															
5	Submission of Progress Report 2															
6	Project Work Continue															
	• Practical/ Laboratory Work															
7	Pre EDX Presentation															
8	Submission of Report Draft															
9	Submission of Soft Cover Dissertation															
10	Final Presentation															

Figure B-2: Semester 2 Gantt chart