NEAR EAST UNIVERSITY

GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

WIRELESS EMERGENCY WARNING SYSTEMS DESIGN AND IMPLEMENTATION

Mehmet Ugurlu

Master Thesis

Department of Computer Engineering

Nicosia-2003

Mehmet Uğurlu: Wireless Emergency Warning Systems Design & Implementation

Approval of the Graduate School of Applied & Social Sciences

Prof. Dr. Fakhraddin Mamedov

Director

We certify that this thesis is satisfactory for the award of the degree of Master of Science in Computer Engineering

Examining Committee in charge:

Prof. Dr. Fakhraddin Mamedov,Committe Chairman.Dean of Engineering Faculty, NEU

Assoc. Prof. Dr. Rahib Abiyev, Committe Member, Computer Engineering Department, NEU

Prof.Dr.Perviz Alizade,Committe Member, Electric &Electronic Engineering, NEU

Assoc. Prof. Dr. Doğan İbrahim,Supervisor, Computer Engineering Department, NEU

ACKNOWLEDGMENTS

I express sincere appreciation to Prof. Dr. Fahreddin Mamedov for his guidence and insight thoroughout the research. Thanks go to the other faculty members, Prof. Dr. Şenol Bektaş for his suggestions and comments. Thanks to Assoc. Prof. Dr. Dogan Ibrahim for his patience during interactive study of my thesis. Also special thanks to Assoc. Prof. Dr. Rahib Abiyev, Research Assistant Aylin Aytaç for her help and to my colleagues.

ABSTRACT

The life is not only becoming easy, but gaining speed also by technological improvements. Especially the wireless technology has become an indispensable factor in communication in which a great advance has been established in last several years. In many fields, people are racing with time and requiring correct information, in appropriate location and time. Administrations are making a loss, since they can't reach information instantly when they need. Mobile phones, Internet, fax-data communication, SMS (short message service) and WAP are becoming widespread in popular fields rapidly. By widening in usage, the GSM (Global System for Mobile Communications) technology is taking part in industrial organizations, hotels, financial associations, residences, media and shopping centers.

The "Wews System" (Wireless Emergency Warning Wystem) that has been developed in this thesis, provides an SMS communication from a PC, fed by sensor signals to mobile phones of the users. The messages include information about failures, dangerous situations and production processes in industrial organizations, residences and shopping centers. In this thesis, information is given on usage of sensors, technical features, montage, electrical and logical connections, pc ports and low level programming and GSM network infrastructure.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Chapter 2

Chapter 4

Chapter 5

Appendix 1

INTRODUCTION

The life is not only becoming easy, but gaining speed also by technological improvements. Especially the wireless technology has become an indispensable factor in communication in which a great advance has been established in last several years. In many fields, people are racing with time and requiring correct information, in appropriate location and time. Administrations are making a loss, since they can't reach information instantly when they need. Mobile phones, Internet, fax-data communication, SMS (short message service) and WAP are becoming widespread in popular fields rapidly. By widening in usage, the GSM (Global System for Mobile Communications) technology is taking part in industrial organizations, hotels, financial associations, residences, media and shopping centers.

The aim of this thesis is to develop a Wireless Emergency Warning System (WEWS) using a standard GSM mobile phone, a PC, and sensors. Basically, the PC receives data from a number of sensors connected to its serial and parallel ports. This data may indicate an emergency, and based on the contents of this data, SMS messages are sent to pre-assigned mobile phones using the GSM technology. This way, an emergency or a warning situation can quickly be transmitted to the interested parties.

There are various studies in world wide spectrum similar to the one in this study. One of these studies is Green House project,which was developed in Denmark by Logic IO Corporation under the unit of Remote Telemetry and Control Units. Another studies are: GSM Alarm Engine project ,which was developed in Deboosere Telecom in Belgium , Palm Size Alarm Monitering System which was developed in Singapora by PQ-Asia Group , The System Ceres modular ,which was developed in United Kingdom by PBE System Corporation ,GSM Engine TC35 which was developed in Germany by Siemens Corporation. The detailed technical properties and configurations of those studies are supplemented in Appendix-1.

The thesis consist of five chapters, conclusion, and two appendixes.

Chapter 1 is about the sensor technology. The characteristics of various industrial and commercial sensors are described in this section in detail.

In the second chapter, the serial and parallel port configurations of standard PCs are outlined and port communication techniques are discussed.

The wireless technology and its usage are described in chapter 3.

Chapter 4 describes the GSM network, GSM technology, SMS message sending techniques, GSM Subscriber Services and the recent advances in the SMS technology.

Chapter 5 is about the wireless emergency warning system designed. Both the hardware and the software details of this system are described in detail. The software uses the standard AT commands and software functions are coded on the Power Builder Programming language. This program checks the sensors and then sends messages to predefined phone numbers using the latest SMS technology.

Finally, information about the standard AT commands and the listing of the program developed are both given in the Appendices.

In conclusion, the results obtained in this thesis indicate the importance of the SMS technology and its usage in the emergency and warning system applications. It is hoped that such systems will soon be commercially available.

CHAPTER 1
SENSORS TECHNOLOGY

1.1 Overview

Microsensors have become an essential element of process control and analytical measurement systems, finding countless applications in, for example, industrial monitoring, factory automation, the automotive industry, transportation, telecommunications, computers and robotics, environmental monitoring, health care, and agriculture; in other words, in almost all spheres of our life. The main driving force behind this progress comes from the evolution in the signal processing. With the development of microprocessors and application-specific integrated circuits (1C), signal processing has become cheap, accurate, and reliable—and it increased the intelligence of electronic equipment. In the early 1980s a comparison in performance/price ratio between microprocessors and sensors showed that sensors were behind. This stimulated research in the sensor area, and soon the race was on to develop sensor technology and new devices. New products and companies have emerged from this effort, stimulating further advances of microsensors. Application of sensors brings new dimensions to products in the form of convenience, energy savings, and safety . Today, we are witnessing an explosion of sensor applications. Sensors can be found in many products, such as microwave and gas ovens, refrigerators, dishwashers, dryers, carpet cleaners, air conditioners, tape recorders, TV and stereo sets, compact and videodisc players. And this is just a beginning.

1.2 Sensor Classification

Sensing the real world requires dealing with physical and chemical quantities that are diverse in nature. From the measurement point of view, all physical and chemical quantities (measurands) can be divided into six signal domains.

The thermal signal domain: the most common signals are temperature, heat, and heat flow.

The mechanical signal domain: the most common signals are force, pressure, velocity, acceleration, and position.

The chemical signal domain: the signals are the internal quantities of the matter such as concentration of a certain material, composition, or reaction rate.

The magnetic signal domain: the most common signals are magnetic field in tensity, flux density, and magnetization.

The radiant signal domain: the signals are quantities of the electromagnetic waves such as intensity, wavelength, polarization, and phase.

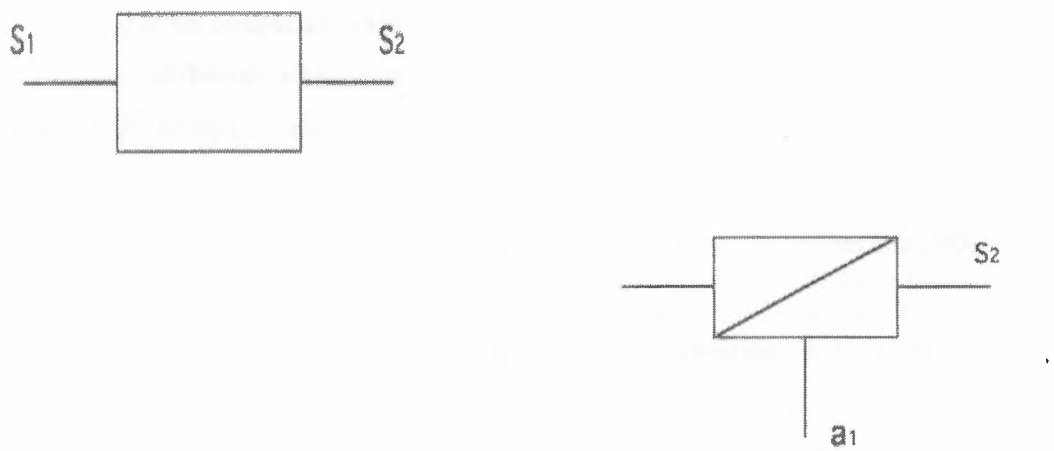
The electrical signal domain: the most common signals are voltage, current, and charge.

As mentioned, sensors convert nonelectrical physical or chemical quantities into electrical signals. It should be also noted that the principle of operation of a particular sensor is dependent on the type of physical quantity it is designed to sense. Therefore, it is no surprise that a general classification of sensors follows the classification of physical quantities. Accordingly, sensors are classified as thermal, mechanical, chemical, magnetic, and radiant.

There is also a classification of sensors based on whether they use an auxiliary energy source or not. Sensors that generate an electrical output signal without an auxiliary energy source are called *self-generating* or *passive.* An example of this type of sensor is a thermocouple. Sensors that generate an electrical output signal with the help of an auxiliary energy source are called *modulating* or *active.* Figure 1.1 shows symbolic presentations of self-generating and modulating sensors. Here, *Si* represents the input signal, s_2 is the output signal, and a, is the auxiliary energy source. In modulating sensors, the auxiliary energy serves as a main source for the output signal, and the measured physical quantity modulates it. This class of sensors

includes magnetotransistors and phototransistors. Modulating sensors are the best choice for the measurement of weak signals.

In addition to the preceding classifications, there are many others based on some common features. A good example is automotive, where



A B

Figure 1.1 Symbolic presentation of self-generating and modulating sensor: (a) self-generating sensor; (b) modulating sensor, where s, is the input signal, s_2 is the output signal, a, is the auxiliary energy source.

the common feature is the application in automobiles for engine and vehicle control. A curious reader can find more information about the classification of sensors in a recently published book on silicon sensors (Ref: **9. Microsensors,MEMS Smart Devices)**

1.3 Sensor Parameters

Performance of sensors, like other electronic devices, is described by parameters.

- Absolute sensitivity is the ratio of the change of the output signal to the change of the measurand (physical or chemical quantity).

- Relative sensitivity is the ratio of a change of the output signal to a change in the measurand normalized by the value of the output signal when the measurand is 0.
- Cross sensitivity is the change of the output signal caused by more than one measurand.
- Direction dependent sensitivity is a dependence of sensitivity on the angle be tween the measurand and the sensor.
- Resolution is the smallest detectable change in the measurand that can cause a change of the output signal.
- Accuracy is the ratio of the maximum error of the output signal to the full-scale output signal expressed in a percentage.
- Linearity error is the maximum deviation of the calibration curve of the output signal from the best fitted straight line that describes the output signal.
- Hysteresis is a lack of the sensor's capability to show the same output signal at a given value of measurand regardless of the direction of the change in the measurand.
    - Offset is the output signal of the sensor when the measurand is 0.
    - Noise is the random output signal not related to the measurand.
- Cutoff frequency is the frequency at which the output signal of the sensor drops to 70.7% of its maximum.
- Dynamic range is the span between the two values of the measurand (maximum and minimum) that can be measured by sensor.
- Operating temperature range is the range of temperature over which the output signal of the sensor remains within the specified error.

It should be pointed out that in addition to these common parameters, other parameters are often used to describe other unique properties of sensors.

1.4 A Seamless Sensor System

Sensing systems are generally used for process control and measurement instrumentation. A simple block diagram of a sensing system is shown in Figure 2 As can be seen, the term *transducer* is used for both the input and the output

blocks of the sensing system. The role of the input transducer is to get information from the real world about a physical or chemical quantity; in other words, to "sense the world." This is the reason why input transducers are commonly called *sensors*. Often the electrical signals generated by sensors are weak and have to be amplified or processed in some way. This is done by the signal processing part of the sensing system. Finally, the role of the output transducer is to convert an electrical signal into a form acceptable for our senses or to initiate some "action," for example, opening or closing a valve. For this reason, output transducers are often called *actuators*. A simple block diagram of the sensing system, as just described, helps to grasp the basic concept of sensing, but it really does not tell the whole story.

Much has been written about the phenomenal development of microelectronics and the strong influence of microprocessors and other integrated circuits on sensing

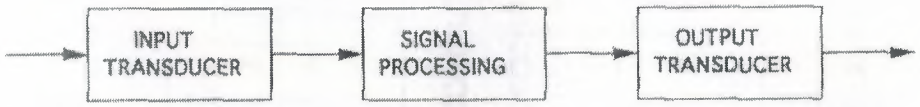


Figure 1. 2 Simple block diagram of the sensing system.

systems. Figure 3 shows a typical sensing system composed of the many devices of modern microelectronics . Following the signal path in Figure 3, one can see that the electrical signals created by sensors are amplified, converted to digital form, and transferred to a microprocessor. The microprocessor also controls a variety of actuators through the interface circuits, where the signals are converted back to analog form and used to drive the actuators. The entire sensing system thus can form a closed control loop.

Also, the microprocessor may communicate with a higher level control computer, making the sensing system, shown in Figure 1.3, part of a larger system. Currently, the type of sensing system shown in Figure 1.3 is spatially distributed and made of

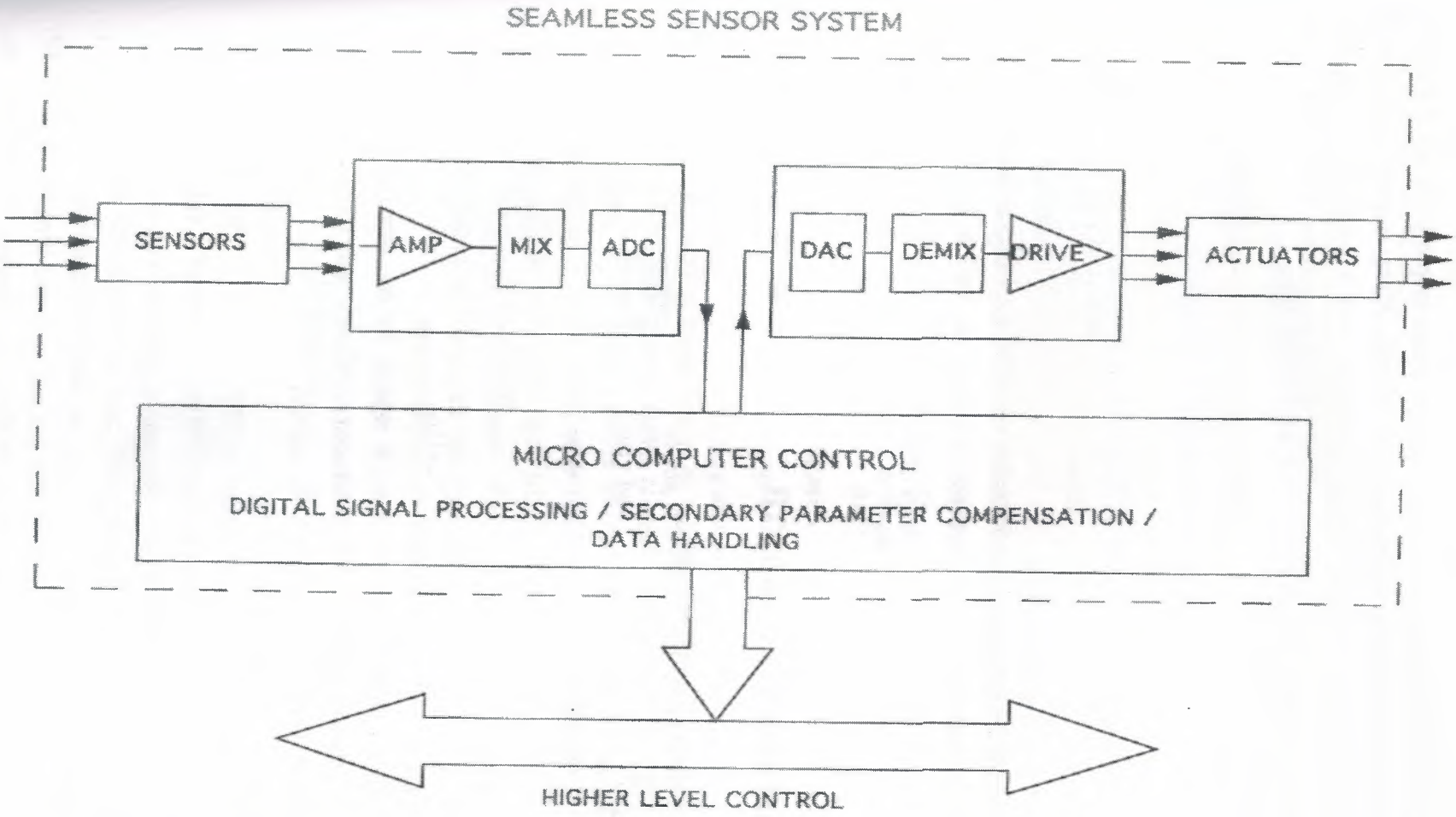


Figure 1.3 Seamless sensor system on the chip.

separate functional blocks. Point-to-point wiring is typically used for the electrical connection between the blocks. Many experts expect in the future that such sensing systems will be integrated into a single chip, forming a "smart" sensor or "seamless" sensor system, where boundaries between the functional blocks will not be apparent.(Ref: **22. Sensors and Control Systems in Manufacturing)**

1.5 Semiconductor Sensor

Semiconductor sensors are transducers that convert mechanical signals into electrical signals. These devices are widely used for the measurement and control of physical variables. Microphones are used in audio systems. Pressure sensors are used in fluidic, pneumatic, and tactile detection systems. Accelerometers are used in navigational and air-bag deployment. Magnetic sensors are used in positional control. Infrared and visible light sensors are used in cameras and night-vision systems. Temperature and flow sensors are used in air conditioning and automotive systems. Chemical sensors are used in biological diagnostic systems. The list of applications of these devices is enormous, and it is growing on a yearly basis. Currently, there is a large demand for low-cost, accurate, and reliable sensors for industrial and consumer product applications.

In the past twenty years, the application of microelectronic technology to the fabrication of mechanical devices greatly stimulated research in semiconductor sensors. Such microfabricated devices are micromachined sensors. Micromachining technology takes advantage of the benefits of semiconductor technology to address the manufacturing and performance requirements of the sensor industry. The versatility of semiconducting materials and the miniaturization of VLSI patterning techniques promise new sensors with better capabilities and improved performance-to-cost ratio over those of conventionally machined devices. Figure 4 shows an example of a microelectromechanical sensing system (MEMS) used in the deployment of air-bags which illustrates the integration of electrical and mechanical devices.

A major factor that contributes to the cost of manufactured products is the overhead expense on production facilities. Technology-based products such as precision electronic and mechanical devices require expensive facilities and highly skilled laborers. These costs are largely independent of the number of products produced. Therefore, the per-unit cost of manufactured goods decreases as the production volume increases. Maximizing throughputs without sacrificing product quality is one of the major goals of manufacturers.

An example that illustrates this point occurs in the microelectronics industry. Integrated-circuit technology allows thousands of electronic circuits to be batch-fabricated simultaneously through a single pass of processing sequences. Batch-fabrication of microelectronic circuits was made possible through the invention of planar technology. In the planar manufacturing process, three-dimensional devices are built on a wafer substrate using stacked layers of planar materials with different but coordinated two-dimensional patterns.

Analog Devices' ADXL-50, the industry's first surface micromachined accelerometer, includes signal conditioning on chip.

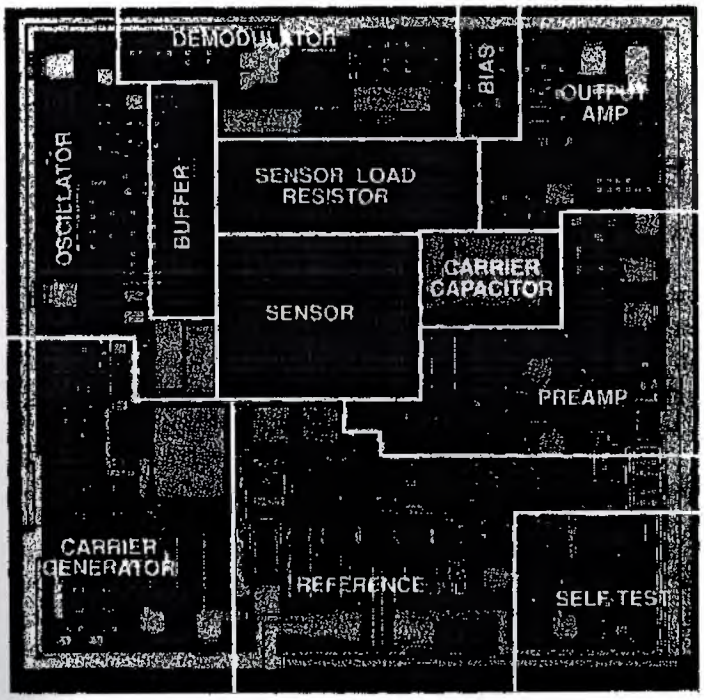


Fig 1.4 State of the art surface micromachined accelerometer that integrates micro-mechanical sensors with BICMOS technology. (Courtesy of Analog Devices.)

By optically repeating the patterns on the wafer, many units are fabricated with just one pass of the process . Micromachined sensors benefit from the same planar manufacturing processes.

Because sensors receptive to different physical variables are structurally different, in general, there is no single technology that allows for the fabrication of a wide variety of sensors. However, there are two major classifications of microsensor technologies. *Bulk-micromachined* sensors are primarily made by the accurate machining of a relatively thick substrate. ***Swface-micrimachined*** sensors are primarily constructed from stacked thin films. Both technologies use materials and processes borrowed from VLSI technology. The three processes of deposition, lithography, and etching are sufficient to construct a wide variety of mechanical structures required for specific sensors. A fundamental sensor-fabrication problem is the development of a suitable fabrication-process sequence of these basic machining steps that define the desired shape and function of the device.(Ref1: **20. Acoustic Wave Sensors: Theory, Design, Physico-Chemical Applications,**Ref2**: 26. Electrochemical Sensors in Immunological Analysis**)

1.6 Sensor Types

a. Acoustic sensors

Acoustic sensors are devices that employ elastic waves at frequencies in the megahertz to low gigahertz range to measure physical, chemical, or biological quantities. Their high sensitivity makes these devices particularly attractive for chemical vapor and gas sensing. In many cases, the output of these sensors is a frequency, which can be measured simply and accurately with an electronic counter. With proper design, these sensors can be quite stable, permitting a \ large dynamic range to be realized.(Ref: **20. Acoustic Wave Sensors: Theory, Design, Physico-Chemical Applications**)

b. Mechanical Semiconductor Sensors

Silicon is used for mechanical sensors, because it combines well-established electronic properties with excellent mechanical properties. Other advantages of silicon include drastically reduced dimensions and mass, batch fabrication and easy interfacing or even integration with electronic circuits and microprocessors. Interest in the mechanical properties of silicon and its use for sensors started with the discovery of its piezoresistivity. The first mechanical sensor was the piezoresistive pressure sensor, but since the development of this sensor, a very wide variety of sensors has been conceived and produced.(Ref: **13. Electromechanical Sensors and Actuators**)

c. Magnetic sensor

A magnetic sensor is capable of converting a magnetic field into a useful electrical / signal. A magnetic sensor is also needed whenever a nonmagnetic signal is / detected by means of an intermediary conversion into a magnetic signal in a so-called tandem transducer. Examples are the detection of a current through its magnetic field or the mechanical displacement of a magnet. Thus, we can distinguish two groups of direct and indirect magnetic-sensor applications.[3]

In *direct* applications, the magnetic sensor is part of a magnetometer. Examples are the measurement of the geomagnetic field, the reading of magnetic data storage media, the identification of magnetic patterns in cards or banknotes, and the control of magnetic apparatus.

In *indirect* applications, the magnetic field is used as an intermediary carrier for detecting nonmagnetic signals. Examples are potential-free current detection for overload protection, integrated watt-hour meters, and contactless linear or angular position, displacement, or velocity detection using a permanent magnet.

These applications require the detection of magnetic fields in the micro- and millitesla range, which can be achieved by integrated semiconductor sensors.

Contactless switching for keyboards or collectorless DC motor control, displacement detection for proximity switches or crankshaft position sensors, and current detection seem to comprise most of the large-scale applications of magnetic sensors. It is for these large-scale applications that inexpensive batch-fabricated semiconductor magnetic

sensors are highly desirable. It is unlikely that integrated silicon magnetic sensors will ever replace nuclear magnetic resonance (NMR) magnetometry with resolution in the nanotesla region, let alone the superconducting quantum interference devices (SQUID) resolving picotesla fields occurring in biomagnetometry.

With respect to the above ranges of magnetic resolution, we recall the following magnetic units. As a measure for the magnetic field strength *H* we use the related magnetic induction *B,* whose unit is 1 tesla = 1 V-s/m^2. This is the inverse of the unit of carrier mobility, namely 1 m^2/V-s = $10^4 cm^2$/V-s=l T. The product of magnetic induction and mobility is a dimensionless number which controls the strength of the galvanomagnetic effects.

Semiconductor magnetic sensors including integrated silicon and GaAs • sensors are useful in the range between 1 /iT and 1 T. Here are some examples II of magnetic induction within that range:

- geomagnetic field 30-60 /*T
- magnetic storage media about 1 mT
- permanent magnets in switches 5-100 mT
- conductor carrying a 10 A current 1 mT
- superconducting coils 10-20 T

.(Refl: **13. Electromechanical Sensors and Actuators,**Ref2: **9. Microsensors,MEMS Smart Devices)**

d. Radiationsensors

Radiation sensors_transform incident radiant signals into standard electrical output signals to be used for data collection,processing and storage.Radiant signals can be categorized into one of the following types: electromagnetic, neutrons, fast electrons, or heavy-charge particles. Electromagnetic radiation and neutrons are uncharged, while fast electrons and heavy-charged particles are charged-particulate radiation.All radiant signals originate in atomic or nuclear processes, and similar techniques are used for their detection.

e. Thermal sensors

The operation of thermal sensors generally can be described in three steps. In the first step the non-thermal quantity is transduced into a thermal quantity by either transducing the power of the non-thermal quantity directly into a heat flow (the self-generating sensors), or by exerting influence by the non-thermal signal on a heat-flow generated by the sensor itself (the modulating sensors). In the second step, the heat flow in the sensor is converted into a temperature difference by means of a thermal resistance. In silicon sensors, micromachining has proved to be a powerful tool for obtaining optimized thermal structures. Closed membranes, cantilever beams and bridges, and floating membranes are often encountered structures in which thermal resistances ahd parallel conductances can be defined very accurately in a simple way. In the third step, the temperature difference is transduced into an electrical signal. The main elements used for this step are transistors or resistors that measure the absolute temperature and are suited for smart sensors, and thermocouples which are interesting for measuring temperature differences, as they can do this without offset and will not spoil the offsetless character of self-generating sensors. .(Ref: **20. Acoustic Wave Sensors: Theory, Design, Physico-Chemical Applications**)

f. Chemical sensors

All the forms of semiconductor chemical sensors have one major problem. In order to detect the chemical species of interest, the sensors must be exposed, unprotected, to the ambient solution or gas. It is difficult to make them reversibly reactive to the gases of interest and nonreactive with respect to all other possible chemical species that may appear in the atmosphere or liquid. Fortunately, in most cases, the form of interference is known and an ideal sensor is not required. For example, the degrading effect of H_2S or $C1_2$ on some sensors is no problem if the user is sure these particular species will not be present.

Sensors from semiconducting metal oxides have the desired feature of low cost, good sensitivity, and convenient form of response (a simple change in resistance). These features have made, and undoubtedly will continue to make, these sensors popular. However, the sensors have problems in reproducibility, stability and selectivity. Every improvement in these aspects will undoubtedly increase the usage of the devices. .(Ref: **20. Acoustic Wave Sensors: Theory, Design, Physico-Chemical Applications**)

g. Biosensor

Biosensors are a special class of chemical sensors that take advantage of the high selectivity and sensitivity of biologically active materials. This high selectivity and sensitivity of the biological material is a result of millions of years of evolution of life on earth, since much of the communication among /biological organisms is based on chemical signals, whether the senses of smell and taste, or immunological reactions, or pheromones, or "hunting" of single-celled organisms. Even the senses of vision, hearing, and touch are transmitted by chemical communication through the nervous system. These communication processes can be considered to be "bio-recognition" processes. Thus, the potential to use these bio-recognition processes as inputs to a sensor is apparent. The diversity of life is reflected in the large variety of biosensors, since there are biological chemicals, organelles, cells, tissues, and organisms that react to everything from small inorganic molecules, such as oxygen, to large, complicated proteins and carbohydrates.(Ref: **8. Sensor Technologies and Data Requirements for ITS Applications**)

CHAPTER 2
PC PORT COMMUNICATIONS

2.1 Overview

Ports are electrical gates which are used for programming peripherals and collecting data from them. There are two types of ports on a standard PC: The Parallel Port, and the Serial Port

Parallel Port is generally used to connect a printer to the PC and this port is easy to program. There are 3 type of pins on a parallel port: Data, Status and Control. The data value of a pin is 1(one) when electrical voltage is aproximatelley +5 Volt at that pin and data value of a pin 0 (zero) when electrical voltage is 0 Volt. The parallel port was chosen in this thesis since most sensors provide analog voltages which can easily be interfaced to the parallel port of a PC using an interface circuitry (look:**Chapter One**) . The data recieved from sensors is faster than other type of ports which provides fast interpretation of data recived from sensors. Disadvantage of the parallel port is that when transmitting data using parallel port the maximum voltage swing is +5V.

Serial Port is used with Universial Asynchronous Reciever Transmitter(UART) circut. There could be one or more serial ports on a standard PC. The data transfer in a serial communication is bit by bit. Another name for this port is Communication port (COM). Serial ports have drawbacks that it is always necessary convert the data into serial code and vice versa.

In this study, the GSM was connected to the serial port of the PC for the following reason:

- UART circut check.
- Bit by bit transfer at serial ports provides sending SMS to multiple predefined phone numbers.
- Further development of serial port is usage infrared, bluetooth devices which immediately proved popular.Many GSM have inbuilt infared, bluetooth devices.Infared and bluetooth technology using serial port communication.
- Serial port communication can be extended to very long distances.

Because of above reasons I chose GSM connect to serial port instead of parallel port.

2.2 Parallel Port

The parallel port is interfaced to the external world using a DB25 (Figure 2.1) type connector with an 8 bit data bus (Pin 2-7), which is used mainly for computer printers.

The standard length of Printer Parallel cables is a maximum of 15 feet although there are 50 foot cables it is not recommended that these cables be used as it can create poor connection and data signals.

2.3 Types Of Parallel Ports

Unidirectional - 4-bit standard port which by factory default did not have the capability of transferring data both ways.

Bi-directional - 8-bit standard port which was released with the introduction of the PS/2 port in 1987 by IBM and are still found in computers today. The Bi-directional port is cable of sending 8-bits input and output. Today on multifunction printers this port can be referred to as a bi-directional, Centronics, PS/2 type or standard port.

EPP - The Enhanced Parallel Port (EPP) was developed in 1991 by Intel, Xircom and Zenith Data Systems and operates close to ISA bus speed and can achieve transfer rates up to 1 to 2MB/sec of data.

EPP version 1.7 released in 1992 and later adapted into the IEEE 1284 standard. All additional features are adapted into the IEEE standard.

EPP version 1.9 never existed.

ECP - The Enhanced Capabilities Port (ECP) was developed by Microsoft and Hewlett-Packard and announced in 1992 is an additional enhanced Parallel port. Unfortunately with ECP it requires an additional DMA channel which can cause resource conflicts.

(Ref: **12. Programming the Parallel Port**)

2.4 Parallel Port Devices

Printer - The most common use for the Parallel port.

Scanner - Another commonly used parallel device is the parallel scanner. Parallel scanners are a popular alternative to SCSI scanners because of how easy they are to to install.

External Drives - Another popular use of the parallel ports are external drives such as sensors and other devices which can be easily removed from one computer and placed onto another.

Port Name	Adress(Hex)	(Hex)
DATA(D)	378	378
STATUS(S)	378+1	379
CONTROL(C)	378+2	37A

Layout

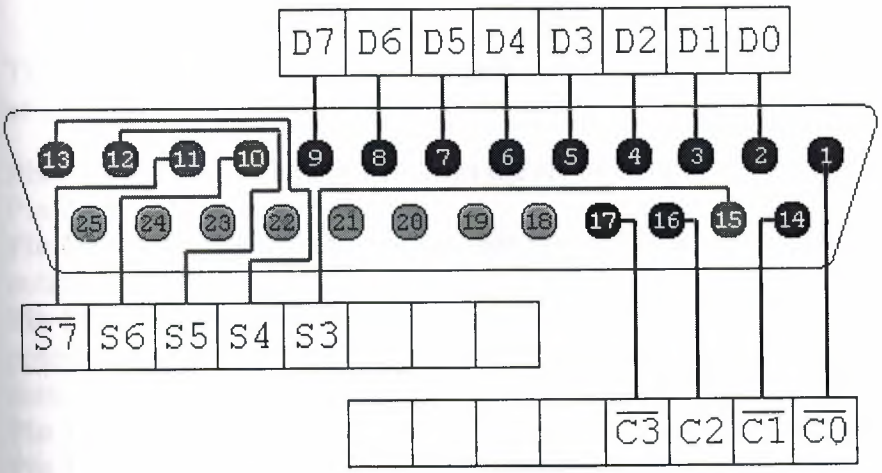


Figure 2.1 DB25 Connector

Signal Name	BIT	PIN	
-Strobe	¬C0	1	Output
+Data Bit 0	D0	2	Output
+Data Bit 1	D1	3	Output
+Data Bit 2	D2	4	Output
+Data Bit 3	D3	5	Output
+Data Bit 4	D4	6	Output
+Data Bit 5	D5	7	Output
+Data Bit 6	D6	8	Output
+Data Bit 7	D7	9	Output
-Acknowledge	S6	10	Input
+Busy	¬S7	11	Input
+Paper End	S5	12	Input
+Select In	S4	13	Input
-Auto Feed	¬C1	14	Output
-Error	S3	15	Input
-Initialize	C2	16	Output
-Select	¬C3	17	Output
Ground	-	18-25	Ground

PIN	PURPOSE
Pin 1	-Strobe
Pin 2	+Data Bit 0
Pin 3	+Data Bit 1
Pin 4	+Data Bit 2
Pin 5	+Data Bit 3
Pin 6	+Data Bit 4
Pin 7	+Data Bit 5
Pin 8	+Data Bit 6
Pin 9	+Data Bit 7
Pin 10	-Acknowledge
Pin 11	+Busy
Pin 12	+Paper End
Pin 13	+Select
Pin 14	-Auto Feed
Pin 15	-Error
Pin 16	-Initialize Printer
Pin 17	-Select Input
Pin 18	-Data Bit 0 Return (GND)
Pin 19	-Data Bit 1 Return (GND)
Pin 20	-Data Bit 2 Return (GND)
Pin 21	-Data Bit 3 Return (GND)
Pin 22	-Data Bit 4 Return (GND)
Pin 23	-Data Bit 5 Return (GND)
Pin 24	-Data Bit 6 Return (GND)
Pin 25	-Data Bit 7 Return (GND)

The following is an explanation of each of the above purposes.

Pin1 = Data acknowledgement when the signal is low.
Pin 2 - 9 = Data transfer pins.
Pin 10 = Acknowledge that the data has finished processing and when the signal is high indicates ready for more.
Pin 11 = When the signal goes high indicate that the printer has accepted the data and is processing it. Once this signal goes low and Pin 10 goes high will accept additional data.
Pin 12 = Printer paper jam when signal is high or no signal if printer jam.
Pin 13 = When high signal printer is indicating that it is on-line and ready to print.
Pin 14 = When low signal PC has indicated that the printer inset a line feed after each line.
Pin 15 = Printer sends data to the computer telling it that an error has occurred.
Pin 16 = When low signal PC has requested that the printer initiate a internal reset.
Pin 17 = When low signal the PC has selected the printer and should in return prepare for data being sent.

Pin 18 - 25 = Ground.(Ref: **17. Parallel Port Complete**)

2.5 Serial Port

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. Usually referred to as a DB9 or DB25 connection both of which adhere to the RS-232c interface standard and defined in ISO 2110 and ISO 4902. D represents the shape of the connector if placed vertically as shown in the below illustrations. The number 9 / 25 indicating the number of pins found on the connector. DB9 Serial connections are now commonly found on modern PC's where DB25 is commonly found on older computers.

2.6 Serial Port Devices

The following is a listing of various hardware components which can be purchased and used with a serial port.

Mouse - One of the most commonly used devices for serial ports, usually used with computers with no PS/2 Ports or laptop computers.

Modem - Another commonly used device for serial ports. Used commonly with older computers however is also commonly used with computers for its ease of use.

Network - One of the original uses of the serial port, which allowed two computers to connect together and allow large files to be transferred between the two.

Printer - Today is not commonly used device for serial ports (not applicable to the DB25 or Parallel Port). However was frequently used with older printers and plotters.

External Drives – In this project a cellular phone is used.

2.7 DB9 Information

The serial port is interfaced to the external world using a serial connector. The most commonly used connector is a 9-pin connector, known as DB9. In the illustration below you can notice several factors to help correctly identify the DB9 Serial connection. First you will notice that the DB9 connection has 9 pins which are each

described in the below chart. The illustration below is an example of the female serial connector which would usually be located on the connector that would connect to the computer. each serial connector generally has two screws measuring .3 cm to allow the serial connection to be securely connected to the back of the computer.

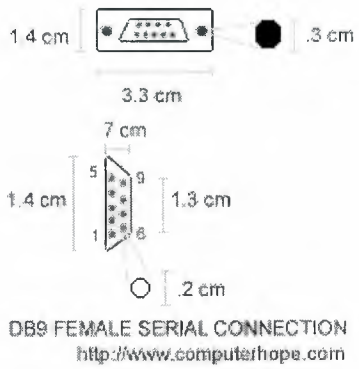


Figure 2.2 DB9 Femail Serial connection

Identifying:

The DB9 serial connection is identified first by its 9 pins.

The DB9 is shaped like a D.

The DB9 will generally be a male connector on the back of the computer.

The following is a listing of each of the pins located on the DB9 connector and what each of these pins are for.

PIN	PURPOSE	SIGNAL NAME
Pin 1	Data Carrier Detect	DCD
Pin 2	Received Data	RxData
Pin 3	Transmitted Data	TxData
Pin 4	Data Terminal Ready	DTR
Pin 5	Signal Ground	Gnd
Pin 6	Data Set Ready	DSR
Pin 7	Request To Send	RTS
Pin 8	Clear To Send	CTS
Pin 9	Ring Indicator	RI

2.8 DB25 Information

DB25 is another type of connector used in serial communications. This is a 25-way connector. In the illustration below you can notice several factors to help correctly identify the DB25 port. First you will notice that the DB25 connection has 25 pins which are each illustrated in the below chart.

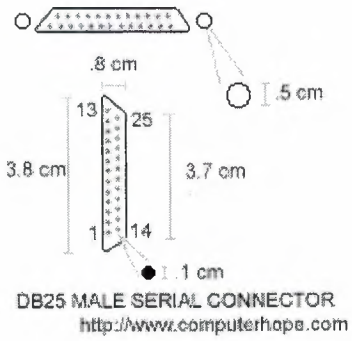


Figure 2.3 Mail Serial Connection

Identifying:

The DB25 serial connection is identified first by its 25 pins.

The DB25 is shaped like a D.

The DB25 is generally be a male connector on the back of the computer.

DB25	Signal Name
8	Data Carrier Detect (DCD)
3	Reive Data (RxD)
2	Transmite Data (TxD)
20	Data Terminal Ready (DTR)
7	Signal Ground (GND)
6	Data Set Ready (DSR)
4	Request to Sent (RTS)
5	Clear to Sent (CTS)
22	Ring Indicator (RI)

Note:1, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 24 and 25 pins are not use in DB25.(Ref: **11. Serial Port Complete**)

CHAPTER 3
WIRELESS CELLULAR PHONE TECHNOLOGY

3.1 Overview

Wireless technologies represent a rapidly emerging area of growth and importance for providing ubiquitous access to the network for all of the campus community. Students, faculty and staff increasingly want un-tethered network access from general-purpose classrooms, meeting rooms, auditoriums, and even the hallways of campus buildings. There is interest in creating mobile computing labs utilizing laptop computers equipped with wireless Ethernet cards. Recently, industry has made significant progress in resolving some constraints to the widespread adoption of wireless technologies. Some of the constraints have included disparate standards, low bandwidth, and high infrastructure and service cost. Wireless technologies can both support the institution mission and provide cost-effective solutions. Wireless is being adopted for many new applications: to connect computers, to allow remote monitoring and data acquisition, to provide access control and security, and to provide a solution for environments where wires may not be the best solution.

3.2 Wireless Technology

Wireless technology uses individual radio frequencies over and over again by dividing a service area into separate geographic zones called cells. Cells can be as small as an individual building (say an airport or arena) or as big as 20 miles across, or any size in between. Each cell is equipped with its own radio transmitter/receiver antenna.

Because the system operates at such a low power, a frequency being used to carry a phone conversation in one cell can be used to carry another conversation in a nearby cell without interference. (this allows much greater capacity than radio systems like Citizens Band (CB) in which all users must try to get their messages on the same limited channels.)

When a customer using a wireless phone - car phone or portable - approaches the boundary of one cell, the wireless network senses that the signal is becoming weak and automatically hands off the call to the antenna in the next cell into which the caller is traveling. When subscribers travel beyond their home geographical area, they can still make wireless calls. The wireless carrier in the area where they are traveling provides the service. This is called roaming

Each cellular antenna is linked to a mobile switching center (MSC), which connects your wireless call to the local "wired" telephone network. Wireless carriers own MSCs.

The mobile telephone industry is limited to 45 megahertz MHz of spectrum bandwidth, which without frequency-reuse, would limit each cellular carrier to 396 frequencies or voice channels. In order to increase calling capacity, these low power facilities "reuse" frequencies on the radio spectrum. The manner in which providers organize, or "configure," their cells is an important factor in increasing frequency reuse and establishing an area's calling capacity.

- Analog cellular operates in the 800MHz frequency range and is available across 95 percent of the United States. Analog cellular service sends a voice through the air using continuous radio waves. As the voice signals travel through the air they get weaker with distance. Equipment in the cellular network returns the signal to its original strength, or amplifies it. This technology is the predominant system in use today. The operating system (called the air interface) for analog is called Advanced Mobile Phone Service (AMPS).

- AMPS stands for advanced mobile phone service. AMPS transmits voices as FM radio signals. The original cellular standard, AMPS is still the most widely used system in the U.S., although digital networks are catching up quickly. A variation on AMPS is narrow-band advanced mobile phone service, or NAMPS, which uses a narrower bandwidth and has greater data capabilities.

- Digital cellular shares the 800MHz frequency band with analog and is usually available where analog service is offered. In digital transmissions, a conversation is converted into the ones and zeros of computer code. Unlike analog transmissions that are sent out as a continuously varying electrical signal in the shape of a wave, digital transmissions are a combination of on-and-off pulses of electricity. Several incompatible air interfaces are used to implement digital cellular networks, including Code Division Multiple Access (CDMA) and Time Division Multiple Access (TDMA).

- CDMA is also known as spread spectrum technology because it uses a low-power signal that is "spread" across a wide bandwidth. With CDMA, a phone call is assigned a code, which identifies it to the correct receiving phone. Using the identifying code and a low-power signal, a large number of calls can be carried simultaneously on the same group of channels.

- TDMA is a digital air interface technology designed to increase the channel capacity by chopping the signal into pieces and assigning each one to a different time slot, each lasting a fraction of a second. Using TDMA, a single channel can be used to handle simultaneous phone calls.

- GSM stands for global system for mobile communications. It is a type of time division multiple access (TDMA) digital wireless network that has encryption features. GSM is rapidly being deployed worldwide and is the standard in Europe at 900MHz. In the U.S., carriers are deploying GSM at 1900MHz, making GSM phones sold in the U.S. incompatible with European GSM phones, and vice versa.

- Personal Communications Service (PCS) is an all-digital service specifically designed for U.S. operations and is available in metropolitan areas. PCS is a term coined by the Federal Communications Commission to describe a digital, two-way, wireless telecommunications system licensed to operate between 1850-1990 MHz, although the FCC's rules describe "PCS" as a broad family of wireless services without reference to spectrum band or technology. PCS is capable of increased call capacity. PCS networks are CDMA, TDMA and global system for mobile communications (GSM).

- Enhanced Specialized Mobile Radio (ESMR) service is also a digital service, formed by the application of digital systems to traditional dispatch "specialized mobile radio" service spectrum, in the 800 and 900 MHz bands. By aggregating this spectrum, and applying a cellular-like digital network, an ESMR company is able to provide a cellular- or PCS-like voice and data messaging service. NEXTEL is one such company, using Motorola's iDEN (TDMA-based) technology to deliver ESMR services in towns and cities across the U.S.

Satellite systems are another viable type of wireless telecommunications service. Instead of sending and receiving signals from a ground-based antenna, wireless phones will communicate via satellites circling the earth.

- GeoSynchronous Satellites: Geosynchronous satellites represent yet another way of providing wireless communications. These satellites, located 22,300 miles above the earth, revolve around the earth once each twenty-four hours-the same as the earth itself. Consequently they appear to be stationary. Communications between two places on earth can take place by using these satellites; one frequency band is used for the uplink, and another for the downlink. Such satellite systems are excellent for the transmission of data, but they leave something to be desired for voice communications. This is a result of the vast distance and the time it takes for an electrical signal to make an earth-satellite-earth round trip. That time amounts to one quarter of a second. A reply from the called subscriber takes another quarter of a second, and the resultant half a second is definitely noticeable. Consequently, voice communications is seldom carried via geosynchronous satellites

Low Earth Orbit (LEO) satellite system. LEOs are satellites that communicate directly with handheld telephones on earth. Because these satellites are relatively low-less than 900 miles-they move across the sky quite rapidly. In a LEO system the communications equipment on a satellite acts much like the cell site of a cellular system. It catches the call from earth and usually passes it to an earth-based switching system. Because of the speed of the satellite, it is frequently necessary to hand off a particular call to a second satellite just rising over the horizon. This is akin

to a cellular system, except that in this case it is the cell site that is moving rather than the subscriber.
(Ref1: 4. The Essential Guide to RF and Wireless,Ref2: 5. 3G Wireless Networks)

3.3 Applications

There are numerous applications for all the different wireless technologies. For the purposes of this paper, applications of wireless technologies are divided into the following:

Voice and messaging,
Hand-held and other Internet-enabled devices, and
Data Networking.

Although a traditional classification, this way of categorizing wireless technologies also includes their differences in cost models, bandwidth, coverage areas, etc. Finally, a section is included on issues related to wireless technologies.(Ref: **21. Introduction to GSM**)

3.4 Voice and Messaging

Cell phones, pagers, and commercial two-way business radios can provide voice and messaging services. These devices may be based on analog or digital standards that differ primarily in the way in which they process signals and encode information. The analog standard is the Advanced Mobile Phone Service (AMPS). Digital standards are Global System for Mobile Communications (GSM), Time Division Multiple Access (TDMA), or Code Division Multiple Access (CDMA).

Normally, devices operate within networks that provide metropolitan, statewide, or nationwide coverage. These large and costly networks are operated by carriers such as, local phone companies, (turkcell,telsim,aycell)etc. Throughput depends on the standard being used, but presently in the Europe, these networks operate throughput rates up to 16 kilobits per second (Kbps). New digital standards, also referred to as

"Third-Generation Services" or 3G, are expected by 2004, and will provide 30 times faster transfer rates and enhanced capabilities. Because of the many standards, there are interoperability issues between networks, carriers, and devices. Generally, charges are based on per minute utilization or per number of messages.
(Ref1: **1. Mobile Messaging Technologies and Services: SMS, EMS and MMS,**Ref2**: 15. GSM and Personal Communications Handbook**)

3.5 Hand-held and Internet-enabled devices

Internet-enabled cell phones and Personal Digital Assistants (PDAs) have emerged as the newest products that can connect to the Internet across a digital wireless network. New protocols, such as Wireless Application Protocol (WAP), and new languages, such as WML (Wireless Markup Language) have been developed specifically for these devices to connect to the Internet. However, the majority of current Internet content is not optimized for these devices; presently, only email, stock quotes, news, messages, and simple transaction-oriented services are available. Other limitations include low bandwidth (less than 14 Kbps), low quality of service, high cost, the need for additional equipment, and high utilization of devices' battery power. Nevertheless, this type of wireless technology is growing rapidly with better and more interoperable products. (Ref:**www.wireless.com**)

3.6 Data Networking - Wireless Local Area Networks(WLAN)

Wireless Local Area Networks (WLAN) are implemented as an extension to wired LANs within a building and can provide the final few meters of connectivity between a wired network and the mobile user.

There are three physical layers for WLANs: two radio frequency specifications (RF' - direct sequence and frequency hopping spread spectrum) and one infrared (IR). Most WLANs operate in the 2.4 GHz license-free frequency band and have throughput rates up to 2 Mbps.

WLAN configurations vary from simple, independent, peer-to-peer connections between a set of PCs, to more complex, intra-building infrastructure networks. There are also point-to-point and point-to-multipoint wireless solutions. A point-to-point solution is used to bridge between two local area networks, and to provide an alternative to cable between two geographically distant locations (up to 30 miles). Point-to-multi-point solutions connect several, separate locations to one single location or building. Both point-to-point and point-to-multipoint can be based on the 802.11b standard or on more costly infrared-based solutions that can provide throughput rates up to 622 Mbps (OC-12 speed). In a typical WLAN infrastructure configuration, there are two basic components:

Access Points - An access point/base station connects to a LAN by means of Ethernet cable. Usually installed in the ceiling, access points receive, buffer, and transmit data between the WLAN and the wired network infrastructure. A single access point supports on average twenty users and has a coverage varying from 20 meters in areas with obstacles (walls, stairways, elevators) and up to 100 meters in areas with clear line of sight. A building may require several access points to provide complete coverage and allow users to roam seamlessly between access points.

Wireless Client Adapter - A wireless adapter connects users via an access point to the rest of the LAN. A wireless adapter can be a PC card in a laptop, an ISA or PCI adapter in a desktop computer, or can be fully integrated within a handheld device.

3.7 Broadband Wireless

Broadband wireless (BW) is an emerging wireless technology that allows simultaneous wireless delivery of voice, data, and video. BW is considered a competing technology with Digital Subscriber Line (DSL). It is generally implemented in metropolitan areas and requires clear line of sight between the transmitter and the receiving end. BW comes in two flavors: Local multi-point

distribution service (LMDS) and Multi-channel multi-point distribution service (MMDS). Both operate in FCC-licensed frequency bands.

LMDS is a high bandwidth wireless networking service in the 28-31 GHz range of the frequency spectrum and has sufficient bandwidth to broadcast all the channels of direct broadcast satellite TV, all of the local over-the-air channels, and high speed full duplex data service. Average distance between LMDS transmitters is approximately one mile apart.

MMDS operates at lower frequencies, in the 2 GHz licensed frequency bands. MMDS has wider coverage than LMDS, up to 35 miles, but has lower throughput rates. Broadband wireless involves costly equipment and infrastructures. However, as it is more widely adopted, it is expected that the service cost will decrease.
(Ref: 25. **Digital Cellular Mobile Radio Links And Networks**)

3.8 Bluetooth

Bluetooth is a technology specification for small form factor, low-cost, short-range wireless links between mobile PCs, mobile phones, and other portable handheld devices, and connectivity to the Internet. The Bluetooth Special Interest Group (SIG) is driving development of the technology and bringing it to market and it includes promoter companies such as 3Com, Ericsson, IBM, Intel, Lucent, Motorola, Nokia, and over 1,800 Adopter/Associate member companies. Bluetooth covers a range of up to ten meters in the unlicensed 2.4GHz band. Because 802.11 WLANs also operate in the same band, there are interference issues to consider. Bluetooth technology and products started being available in 2001, but interoperability seems to be a big problem. By the time and if Bluetooth becomes an adopted technology, current WLANs will already be migrating to the 5 GHz band .
(Ref: **6. Mobile Application Development with SMS and the SIM Toolkit**)

3.9 Important Issues for Wireless

As with any relatively new technology, there are many issues that affect implementation and utilization of wireless networks. There are both common and specific issues depending on the type of wireless network. Some of the common factors include electromagnetic interference and physical obstacles that limit coverage of wireless networks, while others are more specific, such as standards, data security, throughput, ease of use, etc.

Standards

A major obstacle for deployment of wireless networks is the existence of multiple standards. As it was mentioned previously, there are analog and digital standards in wireless telephony. While GSM is the only widely supported standard in Europe and Asia, multiple standards are in use in the U.S. As a result, the U.S. has lagged in wireless networks deployment.Just recently, organizations have been formed to ensure network and device interoperability.

Coverage

Another issue is coverage. Coverage mainly depends on the output power of the transmitter, its location and frequency used to transmit data. For example, lower frequencies are more forgiving when it comes to physical obstacles (walls, stairways, etc.), while high frequencies require clear line of sight. For each particular application, throughput decreases as distance from the transmitter or access point increases.

Security

Data security is a major issue for wireless due to the nature of the transmission mechanism (electromagnetic signals passing through the air). It is commonly believed that voice applications are less secure than data applications. This is due to limited capabilities of existing technologies to protect information that is being transmitted. For example, in metropolitan areas, users are at risk that simple scanning devices can highjack cell phone numbers and be maliciously used. In WLANs,

authentication and encryption provide data security. Current implementations include:

MAC address-based access lists on access points, where only registered and recognized MAC addresses are accepted and allowed to join the network.
A closed wireless system, where users have to know non-advertised the network name to be able to join.

RADIUS server based authentication, where users are authenticated against a centralized RADIUS server based on their MAC address or their username and password.

Wireless Equivalency Privacy (WEP) utilizes data encryption with 40-bit or 128-bit keys that are hidden from users. WEP provides three options, depending on the level of security needed: no encryption of data, combination of encrypted and non-encrypted data, and forced data encryption.
It is important to understand that in WLANs, data is encrypted only between the wireless adapter and the access point. Data travels through a wired LAN unencrypted. Therefore, data transmitted by wireless is not more secure than data transmitted through the wire, but probably not less secure. Application level encryption mechanisms, like secure web transactions (SSL), SSH, etc. are responsible for further protection of data.

CHAPTER 4

GLOBAL SYSTEM FOR MOBILE NETWORK

4.1 Overview

GSM, the Global System for Mobile communications, is a digital cellular communications system which has rapidly gained acceptance and market share worldwide, although it was initially developed in a European context. In addition to digital transmission, GSM incorporates many advanced services and features, including ISDN compatibility and worldwide roaming in other GSM networks. The advanced services and architecture of GSM have made it a model for future third-generation cellular systems, such as UMTS.

The development of GSM started in 1982, when the Conference of European Posts and Telegraphs (CEPT) formed a study group called Groupe Spécial Mobile (the initial meaning of GSM). The group was to study and develop a pan-European public cellular system in the 900 MHz range, using spectrum that had been previously allocated. At that time, there were many incompatible analog cellular systems in various European countries. Some of the basic criteria for their proposed system were:

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
ISDN compatibility

In 1989, the responsibility for GSM was transferred to the European Telecommunication Standards Institute (ETSI). At that time, the United Kingdom requested a specification based on GSM but for higher user densities with low-power mobile stations, and operating at 1.8 GHz. The specifications for this system, called Digital Cellular System (DCS1800) were published 1991. Commercial operation of

GSM networks started in mid-1991 in European countries. By the beginning of 1995, there were 60 countries with operational or planned GSM networks in Europe, the Middle East, the Far East, Australia, Africa, and South America, with a total of over 5.4 million subscribers.There are four GSM operator in our country .

4.2 The GSM Network

GSM provides recommendations, not requirements. The GSM specifications define the functions and interface requirements in detail but do not address the hardware. The reason for this is to limit the designers as little as possible but still to make it possible for the operators to buy equipment from different suppliers. The GSM network is divided into three major systems: the switching system (SS), the base station system (BSS), and the operation and support system (OSS). The basic GSM network elements are shown in *Figure 4.1*. (Ref: **21. Introduction to GSM**)

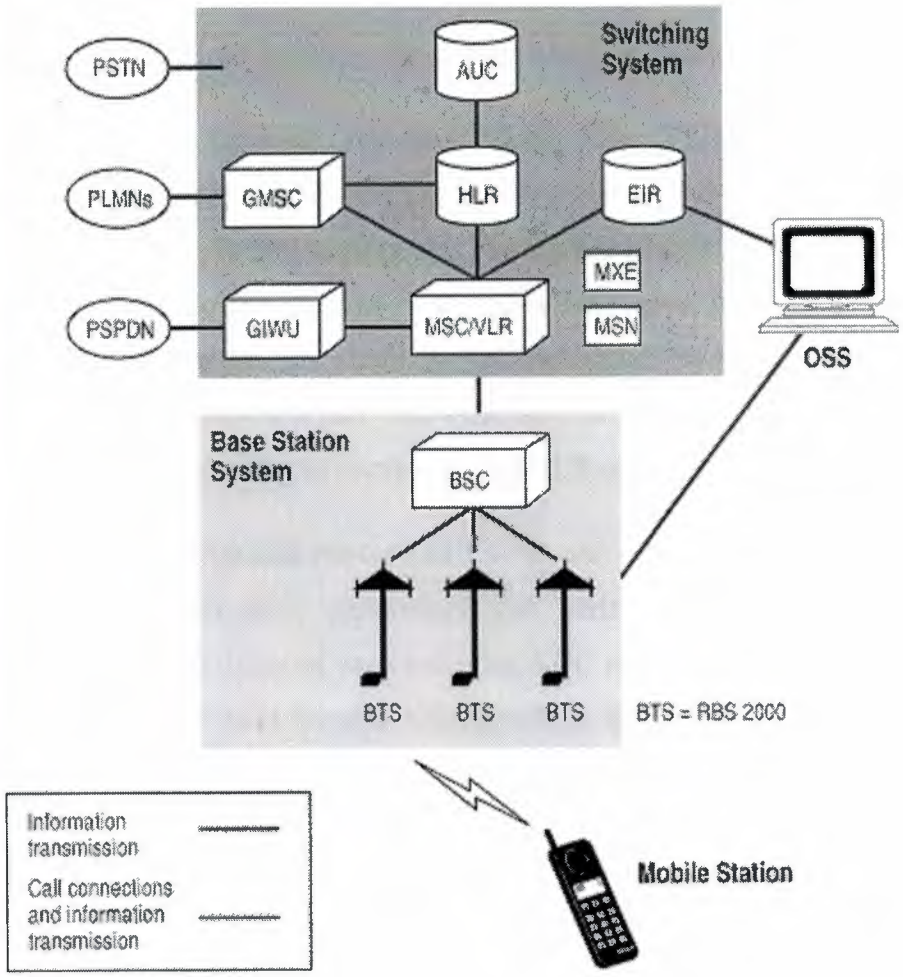


Figure 4.1 GSM Network Elements

4.3 The Switching System

The switching system (SS) is responsible for performing call processing and subscriber-related functions. The switching system includes the following functional units.

- **home location register (HLR)**—The HLR is a database used for storage and management of subscriptions. The HLR is considered the most important database, as it stores permanent data about subscribers, including a subscriber's service profile, location information, and activity status. When an individual buys a subscription from one of the PCS operators, he or she is registered in the HLR of that operator.

- **mobile services switching center (MSC)**—The MSC performs the telephony switching functions of the system. It controls calls to and from other telephone and data systems. It also performs such functions as toll ticketing, network interfacing, common channel signaling, and others.

- **visitor location register (VLR)**—The VLR is a database that contains temporary information about subscribers that is needed by the MSC in order to service visiting subscribers. The VLR is always integrated with the MSC. When a mobile station roams into a new MSC area, the VLR connected to that MSC will request data about the mobile station from the HLR. Later, if the mobile station makes a call, the VLR will have the information needed for call setup without having to interrogate the HLR each time.

- **authentication center (AUC)**—A unit called the AUC provides authentication and encryption parameters that verify the user's identity and ensure the confidentiality of each call. The AUC protects network operators from different types of fraud found in today's cellular world.

- **equipment identity register (EIR)**—The EIR is a database that contains information about the identity of mobile equipment that prevents calls from stolen, unauthorized, or defective mobile stations. The AUC and EIR are implemented as stand-alone nodes or as a combined AUC/EIR node.

(Ref: **25. Digital Cellular Mobile Radio Links And Networks**)

4.4 The Base Station System (BSS)

All radio-related functions are performed in the BSS, which consists of base station controllers (BSCs) and the base transceiver stations (BTSs).

- **BSC**—The BSC provides all the control functions and physical links between the MSC and BTS. It is a high-capacity switch that provides functions such as handover, cell configuration data, and control of radio frequency (RF) power levels in base transceiver stations. A number of BSCs are served by an MSC.
- **BTS**—The BTS handles the radio interface to the mobile station. The BTS is the radio equipment (transceivers and antennas) needed to service each cell in the network. A group of BTSs are controlled by a BSC.(Ref: **19. Mobile Radio Communications**)

4.5 The Operation and Support System

The operations and maintenance center (OMC) is connected to all equipment in the switching system and to the BSC. The implementation of OMC is called the operation and support system (OSS). The OSS is the functional entity from which the network operator monitors and controls the system. The purpose of OSS is to offer the customer cost-effective support for centralized, regional, and local operational and maintenance activities that are required for a GSM network. An important function of OSS is to provide a network overview and support the maintenance activities of different operation and maintenance organizations. (Ref: **2. The GSM Evolution: Mobile Packet Data Services**)

4.6 Additional Functional Elements

Other functional elements shown in *Figure 2* are as follows:

- **message center (MXE)**—The MXE is a node that provides integrated voice, fax, and data messaging. Specifically, the MXE handles short message service, cell broadcast, voice mail, fax mail, e-mail, and notification.
- **mobile service node (MSN)**—The MSN is the node that handles the mobile intelligent network (IN) services.

- **gateway mobile services switching center (GMSC)**—A gateway is a node used to interconnect two networks. The gateway is often implemented in an MSC. The MSC is then referred to as the GMSC.

- **GSM interworking unit (GIWU)**—The GIWU consists of both hardware and software that provides an interface to various networks for data communications. Through the GIWU, users can alternate between speech and data during the same call. The GIWU hardware equipment is physically located at the MSC/VLR.

-

4.7 GSM Network Areas

The GSM network is made up of geographic areas. As shown in *Figure 4.2*, these areas include cells, location areas (LAs), MSC/VLR service areas, and public land mobile network (PLMN) areas.

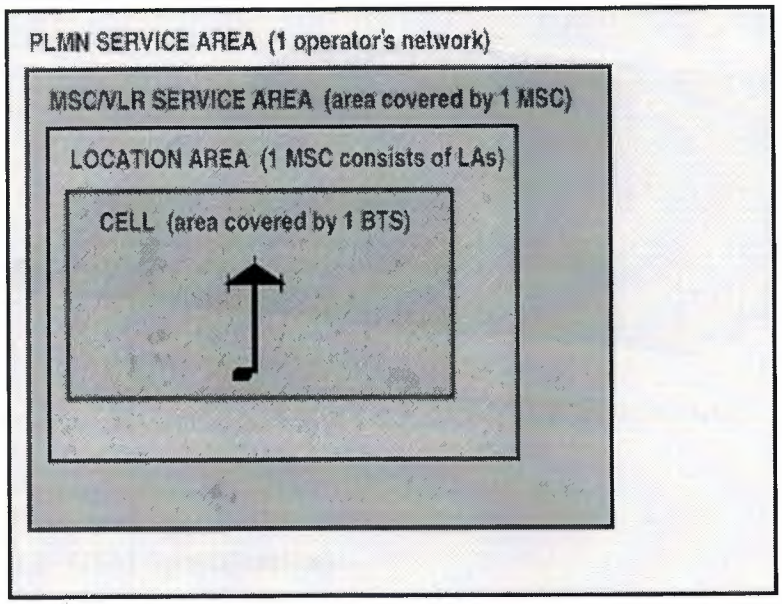


Figure 4.2 Network Areas

The cell is the area given radio coverage by one base transceiver station. The GSM network identifies each cell via the cell global identity (CGI) number assigned to each cell. The location area is a group of cells. It is the area in which the subscriber is paged. Each LA is served by one or more base station controllers, yet only by a single MSC (see *Figure 4.3*). Each LA is assigned a location area identity (LAI) number.

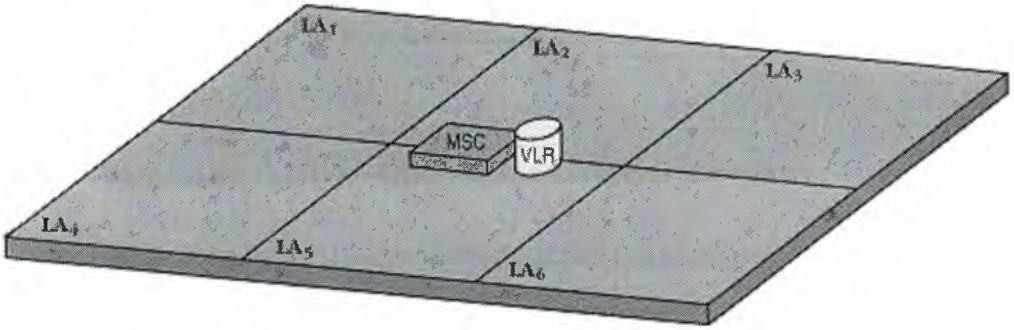


Figure 4.3 Location Areas

An MSC/VLR service area represents the part of the GSM network that is covered by one MSC and which is reachable, as it is registered in the VLR of the MSC (see *Figure 4.4*).

(Ref: **25. Digital Cellular Mobile Radio Links And Networks**)

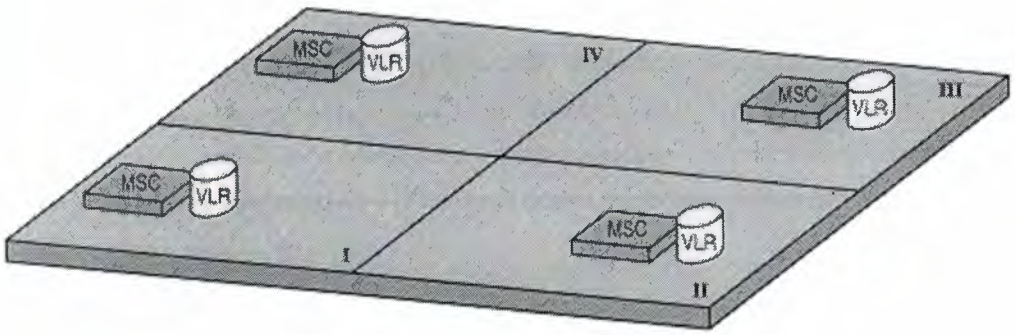


Figure 4.4 MSC/VLR Service Areas

4.8 GSM Specifications

Before looking at the GSM specifications, it is important to understand the following basic terms:

- **bandwidth**—the range of a channel's limits; the broader the bandwidth, the faster data can be sent
- **bits per second (bps)**—a single on-off pulse of data; eight bits are equivalent to one byte

- **frequency**—the number of cycles per unit of time; frequency is measured in hertz (Hz)
- **kilo (k)**—kilo is the designation for 1,000; the abbreviation kbps represents 1,000 bits per second
- **megahertz (MHz)**—1,000,000 hertz (cycles per second)
- **milliseconds (ms)**—one-thousandth of a second
- **watt (W)**—a measure of power of a transmitter

Specifications for different personal communication services (PCS) systems vary among the different PCS networks. Listed below is a description of the specifications and characteristics for GSM.

- **frequency band**—The frequency range specified for GSM is 1,850 to 1,990 MHz (mobile station to base station).
- **duplex distance**—The duplex distance is 80 MHz. Duplex distance is the distance between the uplink and downlink frequencies. A channel has two frequencies, 80 MHz apart.
- **channel separation**—The separation between adjacent carrier frequencies. In GSM, this is 200 kHz.
- **modulation**—Modulation is the process of sending a signal by changing the characteristics of a carrier frequency. This is done in GSM via Gaussian minimum shift keying (GMSK).
- **transmission rate**—GSM is a digital system with an over-the-air bit rate of 270 kbps.
- **access method**—GSM utilizes the time division multiple access (TDMA) concept. TDMA is a technique in which several different calls may share the same carrier. Each call is assigned a particular time slot.
- **speech coder**—GSM uses linear predictive coding (LPC). The purpose of LPC is to reduce the bit rate. The LPC provides parameters for a filter that mimics the

vocal tract. The signal passes through this filter, leaving behind a residual signal. Speech is encoded at 13 kbps.

(Ref1: **27. Advanced Mobile PHOne services:The Cellular Concept,**
Ref2: **21. Introduction to GSM,**
Ref3: **15. GSM and Personal Communications Handbook)**

4.9 GSM Subscriber Services

There are two basic types of services offered through GSM: telephony (also referred to as teleservices) and data (also referred to as bearer services). Telephony services are mainly voice services that provide subscribers with the complete capability (including necessary terminal equipment) to communicate with other subscribers. Data services provide the capacity necessary to transmit appropriate data signals between two access points creating an interface to the network. In addition to normal telephony and emergency calling, the following subscriber services are supported by GSM:

- **dual-tone multifrequency (DTMF)**—DTMF is a tone signaling scheme often used for various control purposes via the telephone network, such as remote control of an answering machine. GSM supports full-originating DTMF.

- **facsimile group III**—GSM supports CCITT Group 3 facsimile. As standard fax machines are designed to be connected to a telephone using analog signals, a special fax converter connected to the exchange is used in the GSM system. This enables a GSM–connected fax to communicate with any analog fax in the network.

- **short message services**—A convenient facility of the GSM network is the short message service. A message consisting of a maximum of 160 alphanumeric characters can be sent to or from a mobile station. This service can be viewed as an advanced form of alphanumeric paging with a number of advantages. If the subscriber's mobile unit is powered off or has left the coverage area, the message is stored and offered back to the subscriber when the mobile is powered on or has reentered the coverage area of the network. This function ensures that the message will be received.

- **cell broadcast**—A variation of the short message service is the cell broadcast facility. A message of a maximum of 93 characters can be broadcast to all mobile subscribers in a certain geographic area. Typical applications include traffic congestion warnings and reports on accidents.

- **voice mail**—This service is actually an answering machine within the network, which is controlled by the subscriber. Calls can be forwarded to the subscriber's voice-mail box and the subscriber checks for messages via a personal security code.

- **fax mail**—With this service, the subscriber can receive fax messages at any fax machine. The messages are stored in a service center from which they can be retrieved by the subscriber via a personal security code to the desired fax number.

(Ref1:**30.www.turkcell.com.tr**,Ref2:**31.www.telsim.com.tr**,Ref3: **1. Mobile Messaging Technologies and Services: SMS, EMS and MMS**)

4.10 Supplementary Services

GSM supports a comprehensive set of supplementary services that can complement and support both telephony and data services. Supplementary services are defined by GSM and are characterized as revenue-generating features. A partial listing of supplementary services follows.

- **call forwarding**—This service gives the subscriber the ability to forward incoming calls to another number if the called mobile unit is not reachable, if it is busy, if there is no reply, or if call forwarding is allowed unconditionally.

- **barring of outgoing calls**—This service makes it possible for a mobile subscriber to prevent all outgoing calls.

- **barring of incoming calls**—This function allows the subscriber to prevent incoming calls. The following two conditions for incoming call barring exist: baring of all incoming calls and barring of incoming calls when roaming outside the home PLMN.

- **advice of charge (AoC)**—The AoC service provides the mobile subscriber with an estimate of the call charges. There are two types of AoC information: one that provides the subscriber with an estimate of the bill and one that can be used for immediate charging purposes. AoC for data calls is provided on the basis of time measurements.

- **call hold**—This service enables the subscriber to interrupt an ongoing call and then subsequently reestablish the call. The call hold service is only applicable to normal telephony.

- **call waiting**—This service enables the mobile subscriber to be notified of an incoming call during a conversation. The subscriber can answer, reject, or ignore the incoming call. Call waiting is applicable to all GSM telecommunications services using a circuit-switched connection.

- **multiparty service**—The multiparty service enables a mobile subscriber to establish a multiparty conversation—that is, a simultaneous conversation between three and six subscribers. This service is only applicable to normal telephony.

- **calling line identification presentation/restriction**—These services supply the called party with the integrated services digital network (ISDN) number of the calling party. The restriction service enables the calling party to restrict the presentation. The restriction overrides the presentation.

- **closed user groups (CUGs)**—CUGs are generally comparable to a PBX. They are a group of subscribers who are capable of only calling themselves and certain numbers.

4.11 The Short Message Service (SMS)

The Short Message Service (SMS) allows text messages to be sent and received to and from mobile telephones. The text can comprise words or numbers or an alphanumeric combination. SMS was created as part of the GSM Phase 1 standard. The first short message is believed to have been sent in December 1992 from a PC to a mobile phone on the Vodafone GSM network in the UK. Each short message is up to 160 characters

in length when Latin alphabets are used, and 70 characters in length when non-Latin alphabets such as Arabic and Chinese are used.

There is no doubting the success of SMS. The market in Europe alone had reached over three billion short messages per month as of December 1999, despite little in proactive marketing by network operators and phone manufacturers. Key market drivers over the next two years, such as the Wireless Application Protocol (WAP), will continue this growth path.

Typical uses of SMS include notifying a mobile phone owner of a voicemail message, alerting a salesperson of an inquiry and telling a driver the address of the next pickup.

4.12 SMS Technology

SMS is essentially similar to paging, but SMS messages do not require the mobile phone to be active and within range, as they will be held for a number of days until the phone is active and within range. SMS messages are transmitted within the same cell or to anyone with roaming capability. They can also be sent to digital phones from a web site equipped with a PC Link or from one digital phone to another. An SMS gateway is a web site that lets you enter an SMS message to someone within the cell served by that gateway or acts as an international gateway for users with roaming capability.

The SMS is a store and forward service. In other words, short messages are not sent directly from sender to recipient, but via an SMS Center. Each mobile telephone network that supports SMS has one or more messaging centers to handle and manage the short messages.

The SMS features confirmation of message delivery. This means that, unlike paging, users do not simply send a short message and trust and hope that it gets delivered. Instead the sender of the short message can receive a return message back notifying them whether the short message has been delivered or not.

Short messages can be sent and received simultaneously with GSM (Global System for Mobile Communications) voice, data and fax calls. This is possible because whereas voice, data and fax calls take over a dedicated radio channel for the duration of the call, short messages travel over and above the radio channel using the signaling path. As

such, users of SMS rarely, if ever, get a busy or engaged signal as they can do during peak network usage times.

Ways of sending multiple short messages are available. SMS concatenation (stringing several short messages together) and SMS compression (getting more than 160 characters of information within a single short message) have been defined and incorporated in the GSM SMS standards.

The network operator needs to purchase its first generation SMS Center as part of the network commissioning plan. The initial SMS Center may simply be a voice mail platform module or a stand-alone SMS Center. It is not possible to make the SMS available without an SMS Center since all short messages pass through the SMS Center.

4.13 Recent SMS Developments

Because simple person-to-person messaging is such an important component of total SMS traffic volumes, anything that simplifies message generation is an important enabler of SMS. Predictive text input algorithms significantly reduce the number of key strokes that need to be made to input a message. T9, from Tegic, anticipates which word the user is trying to generate. Widespread incorporation of such algorithms into the installed base of mobile phones will typically lead to an average uplift in SMS traffic of 25% per enabled user. These predictive text algorithms support multiple languages.

The introduction of standardised protocols such as SIM Application Toolkit and the Wireless Application Protocol (WAP) contribute to an increase in messaging usage by providing a standard service development and deployment environment for application developers and business partners. These protocols also make it easier for users to reply to and otherwise access messaging services through custom menus on the phone. While these protocols are only a means to an end and not new messaging destinations or services, they are likely to lead to a 10-15% uplift in total SMS volumes.

The introduction of more user-friendly terminals contributes to increases in messaging usage. Terminals such as smart phones make it easier for users to originate, reply to and otherwise access messaging services through the provision of a QWERTY keyboard, rather than the limited keypad on standard mobile phones.

(Ref: **1. Mobile Messaging Technologies and Services: SMS, EMS and MMS**)

CHAPTER 5

WIRELESS EMERGENCY WARNING SYTEMS

DESIGN & IMPLEMENTATION

5.1 Overview

In this chapter the development of Wireless Emergency Warning Systems (WEWS) designed by the author is considered. In Figure.5.1 the typical information about system operation is given. We can investigate this figure in 4 stages:

- Sensor computer connection (PCpp)
- WEWS software (PC)
- Computer GSM connection(PCsp)
- GSM Network

Sensors are connected to the computer by the parallel port in the sensor-computer connection. That connects to input pins of LPT port by +5V, input and ground pins. In the event of warning, sensors send a +5V input signal to the PC.

When the software on the PC gets a warning signal, it sends an SMS to predefined user phone numbers at WEWS sysstem table from GSM network by the means of the GSM connection.

System writes the results of this process at c:\wews_result_log file. Computer connects to the serial port (PCsp) as a serial connection with R_x, T_x and groud pins of mobile phone at GSM connection. It then it sends SMS at the direction of AT commands coming from WEWs software. More details on the GSM network were given in Chapter 4.

In this chapter all source code are implemented in Power Builder Programming Language and Database used is Oracle

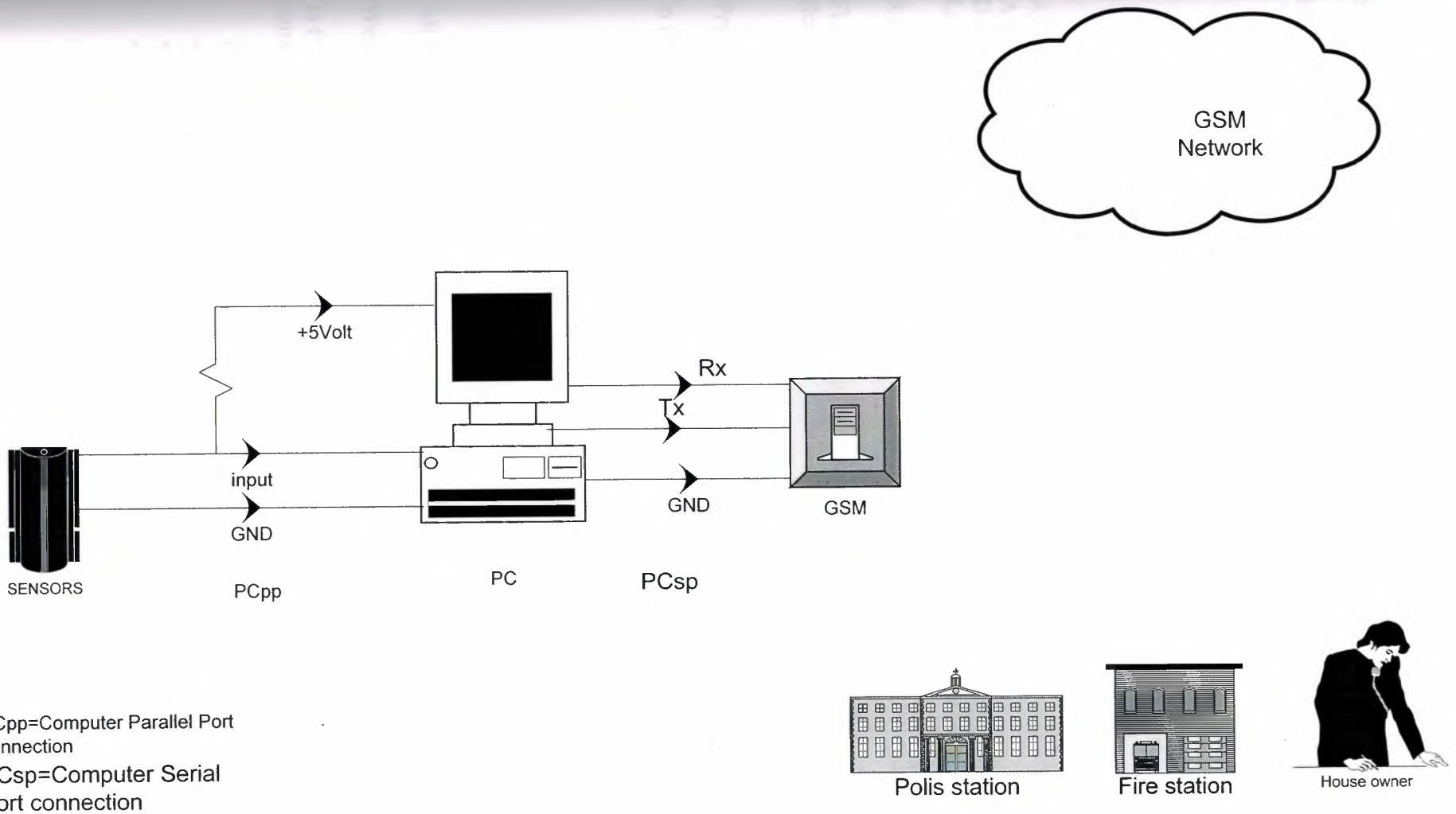


Figure 5.1 Configuration of WEWS

5.2 System Requirements

Minumal required system configurations to run wews is:

- A PC having a microprocessor 80386 or higher
- Necessary space for data at harddisk is 2 MB
- 128Mb or higher RAM
- A GSM ; compatible with ETSI standarts, running at minimal GSME900 Band, having a SMS system
- Sensors

Available Operating Systems MS Windows 95 and higher
A GSM operator that supports SMS
Database with compatible with ANSI SQL(American National Standart Institue Structure Query Language) standarts.In this study ı used Oracle.

5.3 Flowchart of Wews

Flowchart of Wews consist of two stage: one stage is prewews flowchart and other stage is wews flowchart.

Prewews Flowchart

In this section WEWS is prepared by means of executing following process:

- Authtentication and Authorization of system
- Creating the system tables
- Inserting data into system table
- Retrieving data
- Notification of user by Success and Error messages while these processor are occuring.

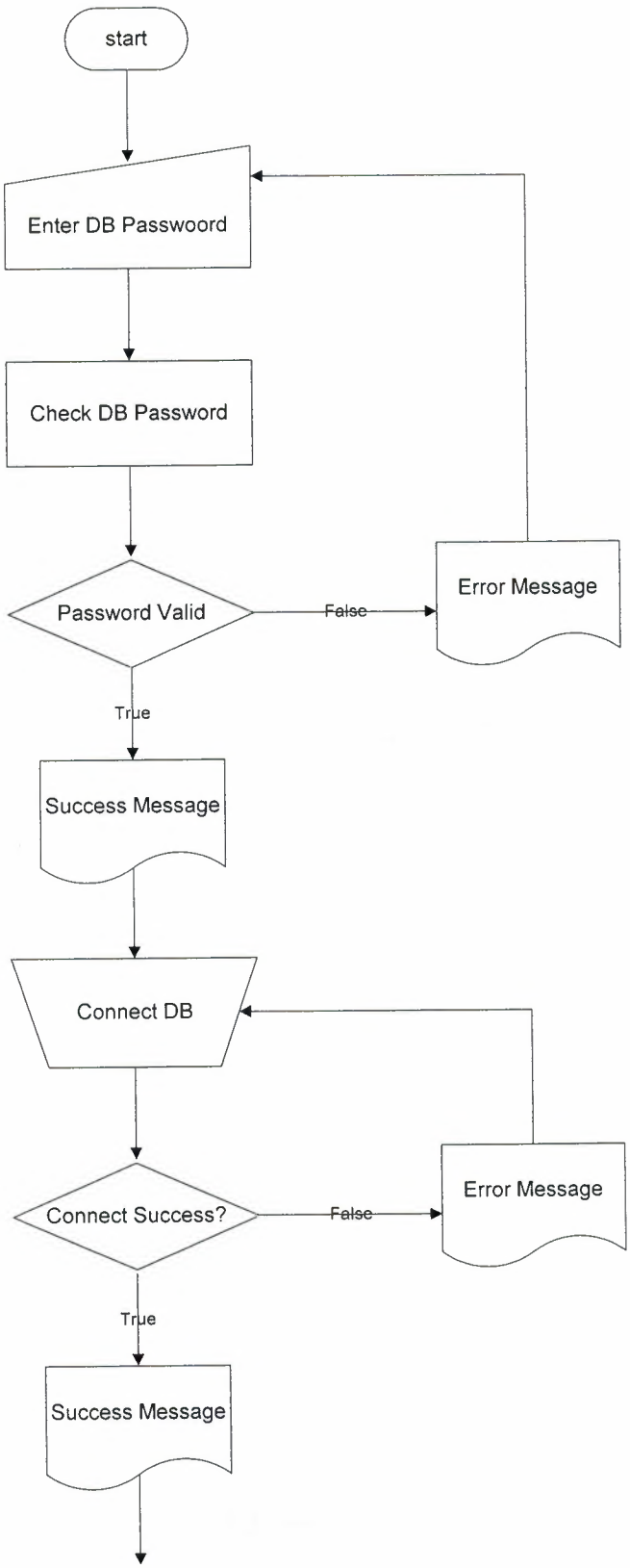
All these processors are shown in Figure 5.2

Wews Flowchart

In Figure 5.3 main bulding blocks and genaral execution logic is depicted in terms of flowchart componenets. There are namely;

- Open ports for communication
- Reading data from ports
- Send SMS processor
- Notification of user by Success and Error messages while these processor are occuring.

Prewews Flowchart



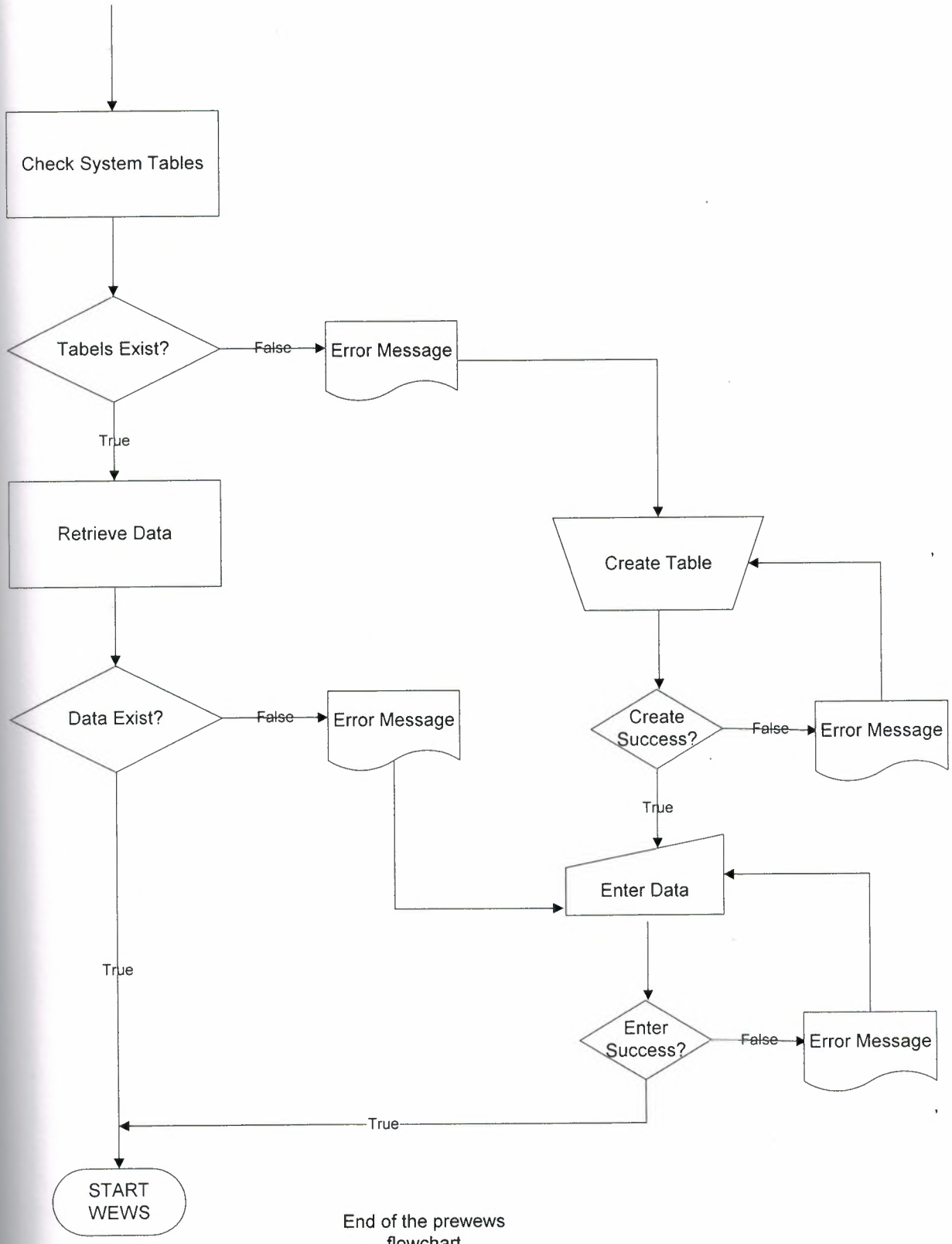
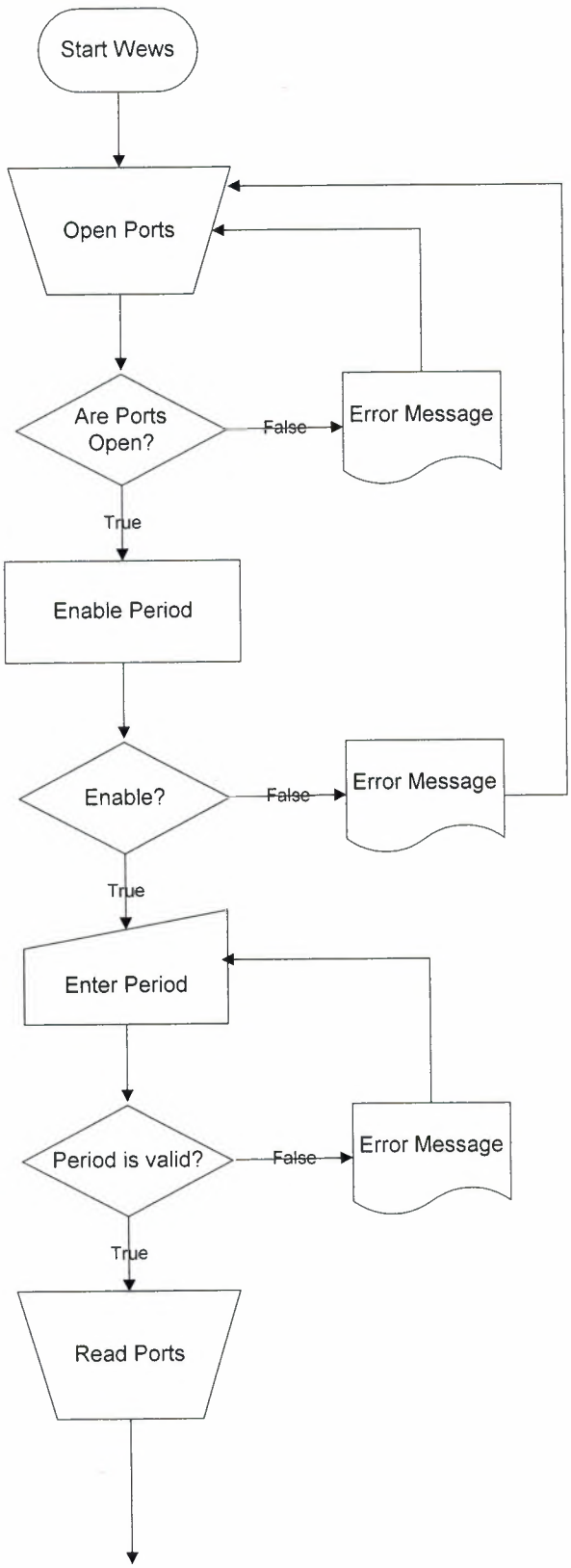


Figure 5.2 Prewews Flowchart

Wews Flowchart



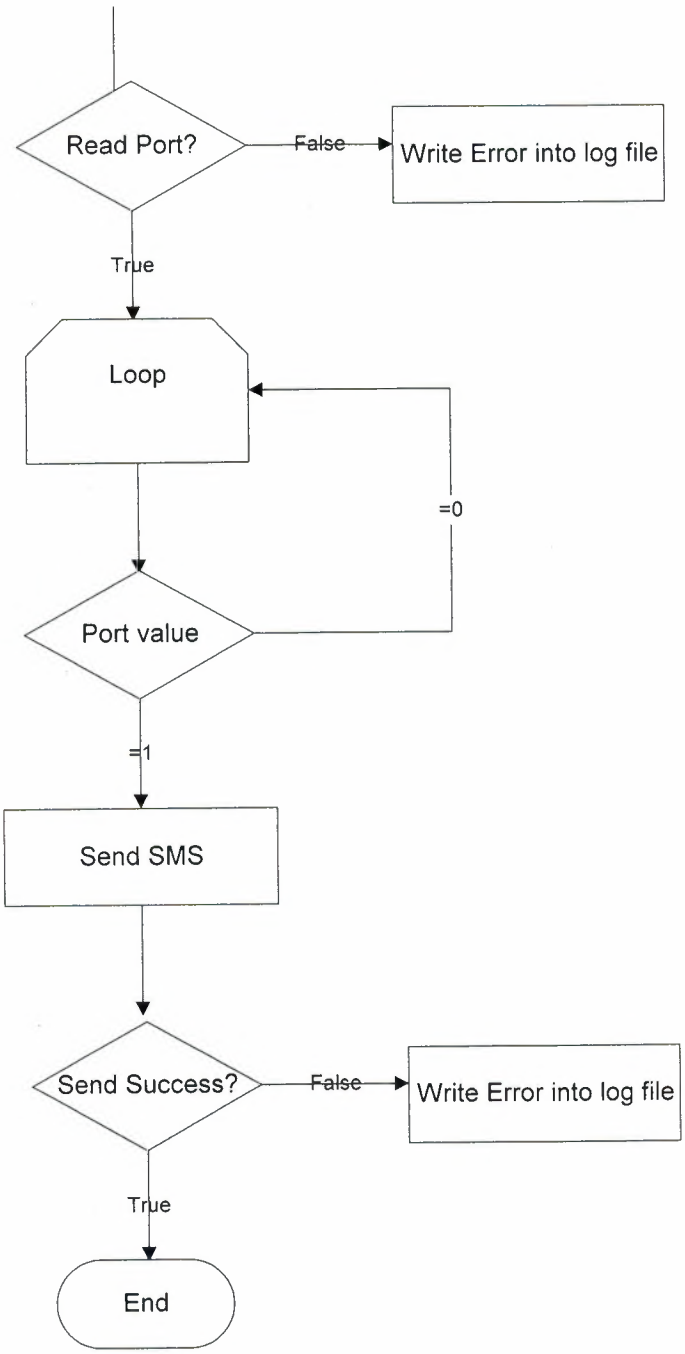


Figure 5.3 Wews Flowchart

5.4 Source Code of Wews

```
//The following source code is external Function for MS windows Api functions.

// External Function Declaration

Function long CreateFile(string lpszName, long fdwAccess, long fdwShareMode, long lpsa, long
fdwCreate, long fdwAttrsAndFlags, long hTemplateFile ) Library "KERNEL32" Alias For
CreateFileA
Function boolean ReadFile(long hFile, ref char lpBuffer[], long nNumberOfBytesToWrite, ref
long lpNumberOfBytesWritten, long lpOverlapped ) Library "KERNEL32"
Function boolean WriteFile(long hFile, ref char lpBuffer[], long nNumberOfBytesToWrite, ref
long lpNumberOfBytesWritten, long lpOverlapped ) Library "KERNEL32"
Function boolean CloseHandle(long hObject ) Library "KERNEL32"

Function boolean SetCommState(long hFile, ref st_dcb lpDCB) Library "KERNEL32"
Function boolean GetCommState(long hFile, ref st_dcb lpDCB) Library "KERNEL32"
Function boolean SetCommTimeouts(long hFile, ref st_commtimeouts lpCommTimeouts)
Library "KERNEL32"
Function boolean GetCommTimeouts(long hFile, ref st_commtimeouts lpCommTimeouts)
Library "KERNEL32"
Function boolean SetupComm(long hFile, long dwInQueue, long dwOutQueue) Library
"KERNEL32"

Subroutine Sleep(long dwMilliseconds) Library "KERNEL32"
Function long GetLastError() Library "KERNEL32"

// Declaration of Related Structures

type st_commtimeouts from structure
unsignedlong readintervaltimeout
unsignedlong readtotaltimeoutmultiplier
unsignedlong readtotaltimeoutconstant
unsignedlong writetotaltimeoutmultiplier
unsignedlong writetotaltimeoutconstant
end type

type st_dcb from structure
unsignedlong dcblength
unsignedlong baudrate
unsignedlong fdummy2
integer wreserved
integer xonlim
integer xofflim
character bytesize
character parity
character stopbits
```

```
End If
Else
CloseHandle(Handle)
RETURN FALSE
End If

If GetCommTimeouts(Handle, lpCTO) Then
//Set up following variables to values you want
lpCTO.ReadIntervalTimeout = Long(50)
lpCTO.ReadTotalTimeoutMultiplier = Long(0)
lpCTO.ReadTotalTimeoutConstant = Long(1000)
lpCTO.WriteTotalTimeoutMultiplier = Long(0)
lpCTO.WriteTotalTimeoutConstant = Long(0)

If Not SetCommTimeouts(Handle, lpCTO) Then
CloseHandle(Handle)
RETURN FALSE
End If
SetupComm(Handle, Long(1024), Long(1024))
Else
CloseHandle(Handle)
RETURN FALSE
End If
Else
RETURN FALSE
End If

RETURN TRUE
end function

//this function's name is f_wite port which is write data to ports .


function long f_writeport (string as_string);
// RETURN VALUES (Example):
// 128 - ERROR
// 131 - SUCCESSFUL
// 100 - PORT DELAY ERROR

Char lch_input[], lch_output[]
Long ll_written = 0, lNull, ll_counter

If Handle <= 0 Then RETURN 100

SetNull(lNull)

ll_counter = Len(as_string)
```

```
Do While ll_counter > 0
lch_input[ll_counter] = Char(Mid(as_string, ll_counter, 1))
ll_counter --
Loop

Sleep(lpcto.ReadIntervalTimeout)

If Not WriteFile(Handle, lch_input, UpperBound(lch_input), ll_written, lNull) Then
RETURN 128 // Write to Port Failed
End If

ll_counter = 5 // 5 Timeouts
lch_output[Len(as_string)] = Char(0) // Initialize output

Do
Sleep(lpcto.ReadIntervalTimeout)
ReadFile(Handle, lch_output, UpperBound(lch_output), ll_written, lNull)
ll_counter --
Loop While ll_written = 0 And ll_counter > 0

Sleep(lpcto.ReadIntervalTimeout)

If ll_counter <= 0 Then RETURN 100 // Port not answered

RETURN 131
end function

//this function's name is f_closeport which is close ports for communication


function boolean f_closeport ();
Boolean b_ret

b_ret = CloseHandle(Handle)
Handle = 0
RETURN b_ret
end function
```

```
character xonchar
character xoffchar
character errorchar
character eofchar
character evtchar
integer wreserved1
end type
// Declaration of Instance Variables
st_dcb lpDCB
st_commtimeouts lpCTO
Long Handle = 0
```
(Ref: **18. Parallel Programming Using C++**)

```
//We create high-level functions to work with. This four functions (f_openport(), f_writeport(),
f_closeport(),f_writeport()) have all to communicate with ports:

//this function's name is f_openport which is open ports for communication

function boolean f_openport (string as_portname);
Long lNull
Long ll_written
String ls_send, ls_property
Constant ULong GENERIC_READ = 2147483648 // (0x80000000L)
Constant ULong GENERIC_WRITE = 1073741824 // (0x40000000L)
Constant ULong OPEN_EXISTING = 3

SetNull(lNull)

// open the port by creating the file
Handle = CreateFile(as_portname, GENERIC_READ + GENERIC_WRITE, 0, lNull,
OPEN_EXISTING, 0, lNull)

If Handle >= 0 Then
If GetCommState(Handle, lpDCB) Then
//BaudRate - 2400,4800,9600,19400,etc.
lpDCB.BaudRate = Long(9600)
//ByteSize(Data Bits) - 4,5,6,7,8
lpDCB.ByteSize = Char(8)
//Parity - 0=None;1=Odd;2=Even;3=Mark;4=Space
lpDCB.Parity = Char(0)
//StopBits - 0=1;1=1.5;2=2
lpDCB.StopBits = Char(0)

If Not SetCommState(Handle, lpDCB) Then
CloseHandle(Handle)
RETURN FALSE
```

```
//this function's name is f_readport which is read data from ports


function long f_readport (string as_portname);
// RETURN VALUES (Example):
// 128 - ERROR
// 131 - SUCCESSFUL
// 100 - PORT DELAY ERROR


Char lch_input[], lch_output[]
Long ll_read = 0, lNull, ll_counter
boolean lb_pos

lb_pos=f_openport(as_portname)
If lb_pos=true then

        If Handle <= 0 Then RETURN 100
        SetNull(lNull)

        If Not ReadFile(Handle, lch_input, UpperBound(lch_input), ll_read, lNull) Then
                RETURN 128    // read from port Failed

        elseif ReadFile(Handle, lch_input, UpperBound(lch_input), ll_read, lNull)=true then
                RETURN ll_read


        End If
else
        RETURN 128
End if



end function
```
(Ref: **24. C Programming Language**)

```
// The purpose of  following function's source code set the window on the center of screen

f_center (window win_name);integer li_width, li_height, li_x, li_y
integer li_maxwidth, li_maxheight
environment le_env

GetEnvironment(le_env)
li_maxwidth = PixelsToUnits(le_env.screenwidth, XPixelsToUnits!)
li_maxheight = PixelsToUnits(le_env.screenheight, YPixelsToUnits!)
li_width = win_name.width
li_height = win_name.height

if win_name.x = 0 then
        li_x = (li_maxwidth - li_width) / 2
        win_name.x = li_x
end if

if win_name.y = 0 then
        li_y = (li_maxheight - li_height - 100) / 2
        win_name.y = li_y
end if
end function


//The following source code is gsm application

global type gsm from application
end type
global transaction sqlca
global dynamicdescriptionarea sqlda
global dynamicstagingarea sqlsa
global error error
global message message
end forward

global variables
st_dcb lpDCB
st_commtimeouts lpCTO
Long Handle = 0
constant string LOG_FILE='C:\WEWS_RESULT.LOG'
end variables

global type gsm from application
string appname = "gsm"
end type
global gsm gsm
```

```
type prototypes
Function long CreateFile(string lpszName, long fdwAccess, long fdwShareMode, long lpsa, long
fdwCreate, long fdwAttrsAndFlags, long hTemplateFile ) Library "KERNEL32" Alias For
CreateFileA
Function boolean ReadFile(long hFile, ref char lpBuffer[], long nNumberOfBytesToWrite, ref
long lpNumberOfBytesWritten, long lpOverlapped ) Library "KERNEL32"
Function boolean WriteFile(long hFile, ref char lpBuffer[], long nNumberOfBytesToWrite, ref
long lpNumberOfBytesWritten, long lpOverlapped ) Library "KERNEL32"
Function boolean CloseHandle(long hObject ) Library "KERNEL32"

Function boolean SetCommState(long hFile, ref st_dcb lpDCB) Library "KERNEL32"
Function boolean GetCommState(long hFile, ref st_dcb lpDCB) Library "KERNEL32"
Function boolean SetCommTimeouts(long hFile, ref st_commtimeouts lpCommTimeouts)
Library "KERNEL32"
Function boolean GetCommTimeouts(long hFile, ref st_commtimeouts lpCommTimeouts)
Library "KERNEL32"
Function boolean SetupComm(long hFile, long dwInQueue, long dwOutQueue) Library
"KERNEL32"

Subroutine Sleep(long dwMilliseconds) Library "KERNEL32"
Function long GetLastError() Library "KERNEL32"
end prototypes

on gsm.create
appname="gsm"
message=create message
sqlca=create transaction
sqlda=create dynamicdescriptionarea
sqlsa=create dynamicstagingarea
error=create error
end on

on gsm.destroy
destroy(sqlca)
destroy(sqlda)
destroy(sqlsa)
destroy(error)
destroy(message)
end on

event open;open(w_wews_data1)
end event
```

```
//The following source code for  wews menu which inherited from PB menu
//menu items are wews,wews log,exit

global type m_wews_data from menu
end type
type m_1 from menu within m_wews_data
end type
type m_viewlog from menu within m_wews_data
end type
type m_exit from menu within m_wews_data
end type
global type m_wews_data from menu
m_1 m_1
m_viewlog m_viewlog
m_exit m_exit
end type
end forward

global type m_wews_data from menu
m_1 m_1
m_viewlog m_viewlog
m_exit m_exit
end type
global m_wews_data m_wews_data

on m_wews_data.create
m_wews_data=this
call super::create
this.m_1=create m_1
this.m_viewlog=create m_viewlog
this.m_exit=create m_exit
this.Item[UpperBound(this.Item)+1]=this.m_1
this.Item[UpperBound(this.Item)+1]=this.m_viewlog
this.Item[UpperBound(this.Item)+1]=this.m_exit
end on

on m_wews_data.destroy
call super::destroy
destroy(this.m_1)
destroy(this.m_viewlog)
destroy(this.m_exit)
end on

type m_1 from menu within m_wews_data
end type

on m_1.create
```

```
call super::create
this.text = "wews"
this.toolbaritemname = "Custom011!"
this.toolbaritemtext = "wews"
end on

on m_1.destroy
call super::destroy
end on

event clicked;open(main)
end event

type m_viewlog from menu within m_wews_data
end type

on m_viewlog.create
call super::create
this.text = "View Log"
this.toolbaritemname = "Application5!"
end on

on m_viewlog.destroy
call super::destroy
end on

event clicked;OpenWithParm(w_log, LOG_FILE)
end event

type m_exit from menu within m_wews_data
end type

event clicked;close(w_wews_data1)
end event
on m_exit.create
call super::create
this.text = "exit"
this.toolbaritemname = "RetrieveCancel!"
this.toolbaritemtext = "exit"
end on

on m_exit.destroy
call super::destroy
end on
```

//The following source code is developed create wews system table those are
Wews and Wews_message.
Description of wews table is

NAME	NOT NULL CHAR(40)
PHONE_NO	NOT NULL CHAR(11)
KIND	NOT NULL CHAR(1)
VALID	NOT NULL CHAR(1)

Description of wews_message table is

KIND	NOT NULL CHAR(1)
MESSAGE_TEXT	NOT NULL CHAR(145)

Also following source code connect database Authentication that the client is who he claims to be using ODBC and retrieves data in addition this source code insert and delete data from database.
//

```
global type w_wews_data1 from window
end type
type mdi_1 from mdiclient within w_wews_data1
end type
type tab_1 from tab within w_wews_data1
end type
type tabpage_1 from userobject within tab_1
end type
type kind_ddlb1 from dropdownlistbox within tabpage_1
end type
type p_del1 from commandbutton within tabpage_1
end type
type p_add1 from commandbutton within tabpage_1
end type
type st_12 from statictext within tabpage_1
end type
type st_11 from statictext within tabpage_1
end type
type st_10 from statictext within tabpage_1
end type
type st_9 from statictext within tabpage_1
end type
type pb1 from commandbutton within tabpage_1
end type
type valid_em1 from editmask within tabpage_1
end type
type phone_em1 from editmask within tabpage_1
```

```
end type
type name_st1 from singlelineedit within tabpage_1
end type
type dw_1 from datawindow within tabpage_1
end type
type gb_4 from groupbox within tabpage_1
end type
type gb_5 from groupbox within tabpage_1
end type
type tabpage_1 from userobject within tab_1
kind_ddlb1 kind_ddlb1
p_del1 p_del1
p_add1 p_add1
st_12 st_12
st_11 st_11
st_10 st_10
st_9 st_9
pb1 pb1
valid_em1 valid_em1
phone_em1 phone_em1
name_st1 name_st1
dw_1 dw_1
gb_4 gb_4
gb_5 gb_5
end type
type tabpage_2 from userobject within tab_1
end type
type kind_ddlb2 from dropdownlistbox within tabpage_2
end type
type m_del2 from commandbutton within tabpage_2
end type
type m_add2 from commandbutton within tabpage_2
end type
type st_14 from statictext within tabpage_2
end type
type st_13 from statictext within tabpage_2
end type
type mb2 from commandbutton within tabpage_2
end type
type mes2 from singlelineedit within tabpage_2
end type
type dw_2 from datawindow within tabpage_2
end type
type gb_3 from groupbox within tabpage_2
end type
type gb_2 from groupbox within tabpage_2
end type
```

```
type tabpage_2 from userobject within tab_1
kind_ddlb2 kind_ddlb2
m_del2 m_del2
m_add2 m_add2
st_14 st_14
st_13 st_13
mb2 mb2
mes2 mes2
dw_2 dw_2
gb_3 gb_3
gb_2 gb_2
end type
type tab_1 from tab within w_wews_data1
tabpage_1 tabpage_1
tabpage_2 tabpage_2
end type
type cb_2 from commandbutton within w_wews_data1
end type
type cb_1 from commandbutton within w_wews_data1
end type
type st_3 from statictext within w_wews_data1
end type
type st_2 from statictext within w_wews_data1
end type
type st_1 from statictext within w_wews_data1
end type
type source from singlelineedit within w_wews_data1
end type
type pas from singlelineedit within w_wews_data1
end type
type us_name from singlelineedit within w_wews_data1
end type
type gb_1 from groupbox within w_wews_data1
end type
end forward

global type w_wews_data1 from window
integer width = 3707
integer height = 2184
boolean titlebar = true
string title = "WEWS INFORMATION WINDOW Mehmet Ugurlu "
string menuname = "m_wews_data"
boolean controlmenu = true
boolean minbox = true
boolean maxbox = true
boolean resizable = true
windowtype windowtype = mdi!
```

```
long backcolor = 67108864
string icon = "Query5!"
mdi_1 mdi_1
tab_1 tab_1
cb_2 cb_2
cb_1 cb_1
st_3 st_3
st_2 st_2
st_1 st_1
source source
pas pas
us_name us_name
gb_1 gb_1
end type
global w_wews_data1 w_wews_data1

type variables
long i_kind_index1,i_kind_index2
end variables
forward prototypes
public subroutine f_dwindow_retrives ()
end prototypes

public subroutine f_dwindow_retrives ();tab_1.tabpage_1.dw_1.SetTransObject(SQLCA)
        tab_1.tabpage_1.dw_1.retrieve()
 tab_1.tabpage_1.dw_1.Modify("DataWindow.ReadOnly=Yes")
 tab_1.tabpage_2.dw_2.SetTransObject(SQLCA)
        tab_1.tabpage_2.dw_2.retrieve()
 tab_1.tabpage_2.dw_2.Modify("DataWindow.ReadOnly=Yes")
end subroutine

on w_wews_data1.create
if this.MenuName = "m_wews_data" then this.MenuID = create m_wews_data
this.mdi_1=create mdi_1
this.tab_1=create tab_1
this.cb_2=create cb_2
this.cb_1=create cb_1
this.st_3=create st_3
this.st_2=create st_2
this.st_1=create st_1
this.source=create source
this.pas=create pas
this.us_name=create us_name
this.gb_1=create gb_1
this.Control[]={this.mdi_1,&
this.tab_1,&
this.cb_2,&
```

```
this.cb_1,&
this.st_3,&
this.st_2,&
this.st_1,&
this.source,&
this.pas,&
this.us_name,&
this.gb_1}
end on

on w_wews_data1.destroy
if IsValid(MenuID) then destroy(MenuID)
destroy(this.mdi_1)
destroy(this.tab_1)
destroy(this.cb_2)
destroy(this.cb_1)
destroy(this.st_3)
destroy(this.st_2)
destroy(this.st_1)
destroy(this.source)
destroy(this.pas)
destroy(this.us_name)
destroy(this.gb_1)
end on

event open;f_center(w_wews_data1)
pas.Password = TRUE
//dw_1.Modify("DataWindow.ReadOnly=Yes")
//dw_2.Modify("DataWindow.ReadOnly=Yes")
////dw_1.selectrow( 1, true)

//m_wews_data menu
//menu=create m_wews_data
//menu.visible=true

end event

type mdi_1 from mdiclient within w_wews_data1
long BackColor=268435456
end type

type tab_1 from tab within w_wews_data1
integer x = 1207
integer y = 224
integer width = 2318
integer height = 1344
integer taborder = 60
```

```
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long backcolor = 67108864
boolean raggedright = true
boolean focusonbuttondown = true
integer selectedtab = 1
tabpage_1 tabpage_1
tabpage_2 tabpage_2
end type

on tab_1.create
this.tabpage_1=create tabpage_1
this.tabpage_2=create tabpage_2
this.Control[]={this.tabpage_1,&
this.tabpage_2}
end on

on tab_1.destroy
destroy(this.tabpage_1)
destroy(this.tabpage_2)
end on

end event

type tabpage_1 from userobject within tab_1
integer x = 18
integer y = 104
integer width = 2281
integer height = 1224
long backcolor = 67108864
string text = "Phone Numbers"
long tabtextcolor = 33554432
long picturemaskcolor = 536870912
kind_ddlb1 kind_ddlb1
p_del1 p_del1
p_add1 p_add1
st_12 st_12
st_11 st_11
st_10 st_10
st_9 st_9
pb1 pb1
valid_em1 valid_em1
phone_em1 phone_em1
```

```
name_st1 name_st1
dw_1 dw_1
gb_4 gb_4
gb_5 gb_5
end type

on tabpage_1.create
this.kind_ddlb1=create kind_ddlb1
this.p_del1=create p_del1
this.p_add1=create p_add1
this.st_12=create st_12
this.st_11=create st_11
this.st_10=create st_10
this.st_9=create st_9
this.pb1=create pb1
this.valid_em1=create valid_em1
this.phone_em1=create phone_em1
this.name_st1=create name_st1
this.dw_1=create dw_1
this.gb_4=create gb_4
this.gb_5=create gb_5
this.Control[]={this.kind_ddlb1,&
this.p_del1,&
this.p_add1,&
this.st_12,&
this.st_11,&
this.st_10,&
this.st_9,&
this.pb1,&
this.valid_em1,&
this.phone_em1,&
this.name_st1,&
this.dw_1,&
this.gb_4,&
this.gb_5}
end on

on tabpage_1.destroy
destroy(this.kind_ddlb1)
destroy(this.p_del1)
destroy(this.p_add1)
destroy(this.st_12)
destroy(this.st_11)
destroy(this.st_10)
destroy(this.st_9)
destroy(this.pb1)
destroy(this.valid_em1)
```

```
destroy(this.phone_em1)
destroy(this.name_st1)
destroy(this.dw_1)
destroy(this.gb_4)
destroy(this.gb_5)
end on

type kind_ddlb1 from dropdownlistbox within tabpage_1
integer x = 1216
integer y = 904
integer width = 411
integer height = 352
integer taborder = 70
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
boolean enabled = false
string text = "none"
boolean sorted = false
string item[] = {"Fire","Thief","Other"}
borderstyle borderstyle = stylelowered!
end type

event selectionchanged;i_kind_index1=index
return i_kind_index1
end event
type p_del1 from commandbutton within tabpage_1
integer x = 1911
integer y = 316
integer width = 183
integer height = 100
integer taborder = 20
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "Delete"
end type

event clicked;string ls_name
ls_name=dw_1.getitemstring(1,'name')
```

```
delete from wews where name=:ls_name;
IF SQLCA.SQLCode = -1 THEN
        MessageBox("WEWS error", 'WEWS TABLE ROW NOT
DELETED'+SQLCA.SQLErrText)

END IF
commit;
dw_1.SetTransObject(SQLCA)
dw_1.retrieve()


end event

type p_add1 from commandbutton within tabpage_1
integer x = 1911
integer y = 132
integer width = 183
integer height = 100
integer taborder = 40
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "Add"
end type

event clicked;kind_ddlb1.enabled=true
name_st1.enabled=true
phone_em1.enabled=true
valid_em1.enabled=true
pb1.enabled=true
end event

type st_12 from statictext within tabpage_1
integer x = 1710
integer y = 828
integer width = 169
integer height = 56
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
```

```
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Valid"
boolean focusrectangle = false
end type

type st_11 from statictext within tabpage_1
integer x = 1257
integer y = 828
integer width = 343
integer height = 56
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Kind"
boolean focusrectangle = false
end type

type st_10 from statictext within tabpage_1
integer x = 782
integer y = 828
integer width = 343
integer height = 56
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Phone No"
boolean focusrectangle = false
end type

type st_9 from statictext within tabpage_1
integer x = 119
integer y = 828
integer width = 197
integer height = 56
integer textsize = -8
```

```
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Name"
boolean focusrectangle = false
end type

type pb1 from commandbutton within tabpage_1
integer x = 1934
integer y = 904
integer width = 137
integer height = 80
integer taborder = 60
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
boolean enabled = false
string text = "..."
end type

event clicked;string ls_kind,ls_name,ls_phone,ls_valid

ls_kind=string(i_kind_index1)
ls_name=name_st1.text
ls_phone=phone_em1.text
ls_valid=valid_em1.text



insert into wews values(:ls_name,:ls_phone,:ls_kind,:ls_valid);

IF SQLCA.SQLCode = -1 THEN
        MessageBox("WEWS error", 'WEWS TABLE NOT INSERTED'+SQLCA.SQLErrText)'

END IF
commit;
tab_1.tabpage_1.dw_1.retrieve()

name_st1.text="
phone_em1.text="
```

```
end type

type name_st1 from singlelineedit within tabpage_1
integer x = 119
integer y = 904
integer width = 617
integer height = 80
integer taborder = 50
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
boolean enabled = false
integer limit = 40
end type

type dw_1 from datawindow within tabpage_1
integer x = 110
integer y = 80
integer width = 1655
integer height = 548
integer taborder = 20
string title = "none"
string dataobject = "dw_data"
boolean vscrollbar = true
boolean resizable = true
boolean livescroll = true
borderstyle borderstyle = styleshadowbox!
end type

event rowfocuschanged;tab_1.tabpage_1.dw_1.SelectRow(0, FALSE)
tab_1.tabpage_1.dw_1.SelectRow(currentrow, TRUE)
end event

type gb_4 from groupbox within tabpage_1
integer x = 46
integer y = 740
integer width = 2153
integer height = 368
integer taborder = 50
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
```

```
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
end type

type gb_5 from groupbox within tabpage_1
integer x = 1819
integer y = 52
integer width = 357
integer height = 404
integer taborder = 40
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
end type

type tabpage_2 from userobject within tab_1
integer x = 18
integer y = 104
integer width = 2281
integer height = 1224
long backcolor = 67108864
string text = "Messages"
long tabtextcolor = 33554432
long picturemaskcolor = 536870912
kind_ddlb2 kind_ddlb2
m_del2 m_del2
m_add2 m_add2
st_14 st_14
st_13 st_13
mb2 mb2
mes2 mes2
dw_2 dw_2
gb_3 gb_3
gb_2 gb_2
end type

on tabpage_2.create
this.kind_ddlb2=create kind_ddlb2
this.m_del2=create m_del2
this.m_add2=create m_add2
```

```
valid_em1.text=''


kind_ddlb1.enabled=false
name_st1.enabled=false
phone_em1.enabled=false
valid_em1.enabled=false
pb1.enabled=false
end event
type valid_em1 from editmask within tabpage_1
integer x = 1682
integer y = 904
integer width = 142
integer height = 80
integer taborder = 60
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
boolean enabled = false
string text = "none"
borderstyle borderstyle = stylelowered!
string mask = "#"
end type

type phone_em1 from editmask within tabpage_1
integer x = 782
integer y = 904
integer width = 398
integer height = 80
integer taborder = 60
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
boolean enabled = false
string text = "none"
borderstyle borderstyle = stylelowered!
maskdatatype maskdatatype = stringmask!
string mask = "0##########"
```

```
this.st_14=create st_14
this.st_13=create st_13
this.mb2=create mb2
this.mes2=create mes2
this.dw_2=create dw_2
this.gb_3=create gb_3
this.gb_2=create gb_2
this.Control[]={this.kind_ddlb2,&
this.m_del2,&
this.m_add2,&
this.st_14,&
this.st_13,&
this.mb2,&
this.mes2,&
this.dw_2,&
this.gb_3,&
this.gb_2}
end on

on tabpage_2.destroy
destroy(this.kind_ddlb2)
destroy(this.m_del2)
destroy(this.m_add2)
destroy(this.st_14)
destroy(this.st_13)
destroy(this.mb2)
destroy(this.mes2)
destroy(this.dw_2)
destroy(this.gb_3)
destroy(this.gb_2)
end on

type kind_ddlb2 from dropdownlistbox within tabpage_2
integer x = 155
integer y = 852
integer width = 411
integer height = 352
integer taborder = 80
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
boolean enabled = false
string text = "none"
```

```
boolean sorted = false
string item[] = {"Fire","Thief","Other"}
end type

event selectionchanged;i_kind_index2=index
return i_kind_index2
end event
type m_del2 from commandbutton within tabpage_2
integer x = 1906
integer y = 360
integer width = 174
integer height = 100
integer taborder = 30
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "Delete"
end type

event clicked;string ls_kind

ls_kind=dw_2.getitemstring(1,'kind')


delete from wews_message where kind=:ls_kind;

IF SQLCA.SQLCode = -1 THEN
        MessageBox("WEWS error", 'WEWS_MESSAGE TABLE ROW NOT
DELETED'+SQLCA.SQLErrText)

END IF
commit;
tab_1.tabpage_2.dw_2.SetTransObject(SQLCA)
tab_1.tabpage_2.dw_2.retrieve()

end event

type m_add2 from commandbutton within tabpage_2
integer x = 1920
integer y = 160
integer width = 160
integer height = 100
integer taborder = 50
integer textsize = -8
```

```
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "Add"
end type

event clicked;kind_ddlb2.enabled=true
mes2.enabled=true
mb2.enabled=true
end event

type st_14 from statictext within tabpage_2
integer x = 681
integer y = 772
integer width = 366
integer height = 56
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Message Text"
boolean focusrectangle = false
end type

type st_13 from statictext within tabpage_2
integer x = 155
integer y = 772
integer width = 343
integer height = 56
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Kind"
boolean focusrectangle = false
end type
```

```
type mb2 from commandbutton within tabpage_2
integer x = 1554
integer y = 844
integer width = 151
integer height = 88
integer taborder = 80
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
boolean enabled = false
string text = "..."
end type

event clicked;string ls_kind,ls_text

ls_kind=string(i_kind_index2)
ls_text=mes2.text

insert into  wews_message values(:ls_kind,:ls_text);

IF SQLCA.SQLCode = -1 THEN
        MessageBox("WEWS error", 'WEWS_MESSAGE TABLE NOT
INSERTED'+SQLCA.SQLErrText)

END IF
commit;
tab_1.tabpage_2.dw_2.retrieve()

kind_ddlb2.enabled=false
mes2.enabled=false
mes2.text=''
mb2.enabled=false
end event
type mes2 from singlelineedit within tabpage_2
integer x = 681
integer y = 852
integer width = 786
integer height = 80
integer taborder = 70
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
```

```
string facename = "Arial"
long textcolor = 33554432
boolean enabled = false
integer limit = 125
end type

type dw_2 from datawindow within tabpage_2
integer x = 101
integer y = 88
integer width = 1664
integer height = 572
integer taborder = 30
string title = "none"
string dataobject = "dw_message"
boolean hscrollbar = true
boolean vscrollbar = true
borderstyle borderstyle = styleshadowbox!
end type

event rowfocuschanged;tab_1.tabpage_2.dw_2.SelectRow(0, FALSE)
tab_1.tabpage_2.dw_2.SelectRow(currentrow, TRUE)
end event

type gb_3 from groupbox within tabpage_2
integer x = 69
integer y = 724
integer width = 1810
integer height = 304
integer taborder = 60
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
end type

type gb_2 from groupbox within tabpage_2
integer x = 1833
integer y = 92
integer width = 379
integer height = 444
integer taborder = 40
integer textsize = -8
integer weight = 400
```

```
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
borderstyle borderstyle = stylelowered!
end type

type cb_2 from commandbutton within w_wews_data1
integer x = 681
integer y = 832
integer width = 343
integer height = 84
integer taborder = 50
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "Create Table"
end type

event clicked;string ls_sql,ls_sql2
ls_sql="CREATE TABLE WEWS (name            char(40)            Primary Key,"&
+"phone_no          Char(11)            Not Null, Kind      char(1)    Not Null,"&
+"Valid      char(1)    Not Null)"

EXECUTE IMMEDIATE :ls_sql ;

IF SQLCA.SQLCode = 0 THEN
        MessageBox("WEWS", 'TABLE CREATED')
ELSEIF SQLCA.SQLCode = -1 THEN
        MessageBox("WEWS error", 'TABLE NOT CREATED'+SQLCA.SQLErrText)

END IF

ls_sql2="CREATE TABLE WEWS_MESSAGE (Kind            char(1)         Primary Key,"&
+"message_text          Char(145)           Not Null)"

EXECUTE IMMEDIATE :ls_sql2 ;


IF SQLCA.SQLCode = 0 THEN
        MessageBox("WEWS", 'WEWS_MESSAGE TABLE CREATED')
ELSEIF SQLCA.SQLCode = -1 THEN
```

```
        MessageBox("WEWS error", 'WEWS_MESSAGE TABLE NOT
CREATED'+SQLCA.SQLErrText)

END IF
end event

type cb_1 from commandbutton within w_wews_data1
integer x = 681
integer y = 700
integer width = 343
integer height = 84
integer taborder = 40
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "Connect DB"
end type

event clicked;string ls_user,ls_pasword,ls_source
boolean lb_allports=false
lb_allports=f_closeport()


ls_user=us_name.text
ls_pasword=pas.text
ls_source=source.text

SQLCA.DBMS = "ODBC"
SQLCA.AutoCommit = False
SQLCA.DBParm =
"ConnectString='DSN="+ls_source+";UID="+ls_user+";PWD="+ls_pasword+""""

Connect using SQLCA;

IF SQLCA.SQLCode = 0 THEN
        MessageBox("WEWS", 'DB CONNECTION IS SUCCESS')
        f_dwindow_retrives()

ELSEIF SQLCA.SQLCode = -1 THEN
        MessageBox("WEWS error", 'DB CONNECTION NOT
SUCCESS'+SQLCA.SQLErrText)

END IF
```

```
end event

type st_3 from statictext within w_wews_data1
integer x = 160
integer y = 572
integer width = 343
integer height = 56
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Source Name"
alignment alignment = right!
boolean focusrectangle = false
end type

type st_2 from statictext within w_wews_data1
integer x = 160
integer y = 444
integer width = 343
integer height = 56
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Password"
alignment alignment = right!
boolean focusrectangle = false
end type

type st_1 from statictext within w_wews_data1
integer x = 160
integer y = 316
integer width = 343
integer height = 56
```

```
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "User ID"
alignment alignment = right!
boolean focusrectangle = false
end type

type source from singlelineedit within w_wews_data1
integer x = 512
integer y = 572
integer width = 539
integer height = 80
integer taborder = 30
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
borderstyle borderstyle = stylelowered!
end type

type pas from singlelineedit within w_wews_data1
integer x = 512
integer y = 436
integer width = 539
integer height = 80
integer taborder = 20
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
borderstyle borderstyle = stylelowered!
end type

type us_name from singlelineedit within w_wews_data1
integer x = 517
```

```
integer y = 300
integer width = 539
integer height = 80
integer taborder = 10
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
borderstyle borderstyle = stylelowered!
end type

type gb_1 from groupbox within w_wews_data1
integer x = 78
integer y = 196
integer width = 1038
integer height = 776
integer textsize = -8
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "DB "
borderstyle borderstyle = stylelowered!
end type
```

```
//
The following source code is developed for scan port and  sending  SMS to predefined user
phone number and message text at wews system table from GSM network by the means of the
computer  GSM connection and writing results at the  text file(C:\WEWS_RESULT.LOG)
In this code using AT Commands and SMS functions  (ETSI GSM 07.05)
//

global type main from window
end type
type hpb_1 from hprogressbar within main
end type
type cb_3 from commandbutton within main
end type
type per_em from editmask within main
end type
type st_2 from statictext within main
end type
type st_1 from statictext within main
end type
type cb_2 from commandbutton within main
end type
type cb_1 from commandbutton within main
end type
type dw_1 from datawindow within main
end type
type gb_1 from groupbox within main
end type
type gb_2 from groupbox within main
end type
type gb_3 from groupbox within main
end type
type gb_4 from groupbox within main
end type
type gb_5 from groupbox within main
end type
end forward

global type main from window
integer width = 2437
integer height = 1604
boolean titlebar = true
string title = "WEWS Mehmet Ugurlu"
boolean controlmenu = true
windowtype windowtype = response!
```

```
long backcolor = 67108864
hpb_1 hpb_1
cb_3 cb_3
per_em per_em
st_2 st_2
st_1 st_1
cb_2 cb_2
cb_1 cb_1
dw_1 dw_1
gb_1 gb_1
gb_2 gb_2
gb_3 gb_3
gb_4 gb_4
gb_5 gb_5
end type
global main main

type variables

end variables

on main.create
this.hpb_1=create hpb_1
this.cb_3=create cb_3
this.per_em=create per_em
this.st_2=create st_2
this.st_1=create st_1
this.cb_2=create cb_2
this.cb_1=create cb_1
this.dw_1=create dw_1
this.gb_1=create gb_1
this.gb_2=create gb_2
this.gb_3=create gb_3
this.gb_4=create gb_4
this.gb_5=create gb_5
this.Control[]={this.hpb_1,&
this.cb_3,&
this.per_em,&
this.st_2,&
this.st_1,&
this.cb_2,&
this.cb_1,&
this.dw_1,&
this.gb_1,&
this.gb_2,&
this.gb_3,&
this.gb_4,&
```

```
this.gb_5}
end on

on main.destroy
destroy(this.hpb_1)
destroy(this.cb_3)
destroy(this.per_em)
destroy(this.st_2)
destroy(this.st_1)
destroy(this.cb_2)
destroy(this.cb_1)
destroy(this.dw_1)
destroy(this.gb_1)
destroy(this.gb_2)
destroy(this.gb_3)
destroy(this.gb_4)
destroy(this.gb_5)
end on

event timer;long ll_pp,ll_result,ll_fp
string ls_message_text,ls_name,ls_phoneno,ls_message

st_2.text=String(Now(), "hh:mm:ss")
hpb_1.offsetpos(1)
//AT+CMGF=0    Set PDU mode

//AT commans errors

//0 OK
//300  Phone failure
//
//301  SMS service of phone reserved
//
//302  Operation not allowed
//
//303  Operation not supported
//
//304  Invalid PDU mode parameter
//
//305  Invalid text mode parameter
//
//310  SIM not inserted
//
//311  SIM PIN necessary
//
//312  PH-SIM PIN necessary
//
```

```
//313  SIM failure
//
//314  SIM busy
//
//315  SIM wrong
//
//320  Memory failure
//
//321  Invalid memory index
//
//322  Memory full
//
//330  SMSC (message service center) address unknown
//
//331  No network service
//
//332  Network timeout
//
//500  Unknown error
//
//512  Manufacturer specific
//
//



select message_text into :ls_message_text from wews_message where kind='3';

ll_pp=f_readport('COM1')

if ll_pp=131 then

                declare pnumbers cursor for
                select name,phone_no from wews
                where kind='3';

                open pnumbers;
                fetch pnumbers into:ls_name,:ls_phoneno;

                DO WHILE (sqlca.sqlcode=0)
                        ls_message='urgent'+ls_name+ls_message_text


                                ll_result=f_writeport('AT+CMGF=0
AT+CMGS='+ls_phoneno+ls_message)
                                if not(isnull(ll_result)) then
```

```
        ll_fp=FileOpen(LOG_FILE,LineMode!,Write!,LockReadWrite!,Replace!)
                                                if ll_fp <> -1 then
                                                        FileWrite(ll_fp, string(ll_result)+'
'+string(Today(), "dd/mm/yy hh:mm:ss"))
                                                end if
                                        end if
                                        fileclose(ll_fp)


                                fetch pnumbers into:ls_phoneno;
                        LOOP

    close pnumbers;
elseif ll_pp=128 then
        ll_fp=FileOpen(LOG_FILE,LineMode!,Write!,LockReadWrite!,Replace!)
                if ll_fp <> -1 then
                        FileWrite(ll_fp, string(ll_pp)+' PORT ERROR '+string(Today(),
"dd/mm/yy hh:mm:ss"))
                end if
        fileclose(ll_fp)
elseif ll_pp=100 then
        ll_fp=FileOpen(LOG_FILE,LineMode!,Write!,LockReadWrite!,Replace!)
                if ll_fp <> -1 then
                        FileWrite(ll_fp, string(ll_pp)+' PORT DELAY ERROR '+string(Today(),
"dd/mm/yy hh:mm:ss"))
                end if
        fileclose(ll_fp)
end if

end event

event open;     f_center(main)
        dw_1.SetTransObject(SQLCA)
        dw_1.retrieve()
        dw_1.Modify("DataWindow.ReadOnly=Yes")

end event

type hpb_1 from hprogressbar within main
integer x = 137
integer y = 1420
integer width = 2222
integer height = 64
unsignedinteger minposition = 50
unsignedinteger maxposition = 1000
unsignedinteger position = 50
```

```
integer setstep = 2
end type

type cb_3 from commandbutton within main
integer x = 2002
integer y = 1132
integer width = 215
integer height = 88
integer taborder = 40
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "Exit"
end type

event clicked;close(main)
end event

type per_em from editmask within main
integer x = 1399
integer y = 532
integer width = 123
integer height = 72
integer taborder = 20
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
boolean enabled = false
string text = "none"
borderstyle borderstyle = stylelowered!
string mask = "##"
end type

type st_2 from statictext within main
integer x = 1861
integer y = 528
integer width = 334
integer height = 84
integer textsize = -8
integer weight = 700
```

```
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
alignment alignment = right!
boolean focusrectangle = false
end type

type st_1 from statictext within main
integer x = 224
integer y = 68
integer width = 599
integer height = 56
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Urgent Phone Numbers"
borderstyle borderstyle = styleshadowbox!
boolean focusrectangle = false
end type

type cb_2 from commandbutton within main
integer x = 2002
integer y = 868
integer width = 215
integer height = 88
integer taborder = 30
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
boolean enabled = false
string text = "okay"
end type

event clicked;integer li_period
li_period=integer(per_em.text)
hpb_1.offsetpos( 2)
```

```
//long i
//i=0
//do while true
//
//
// i++
//        st_2.text=string(i)
//        IF i= 20000 THEN EXIT
//        this.cancel=true
//loop
//
//if ll > 0 then
//        messagebox('',string(ll))
//end if


if li_period=0 then
        messagebox('WEWS','Please Enter Period')
else
        timer(li_period)

end if
end event

type cb_1 from commandbutton within main
integer x = 2002
integer y = 292
integer width = 215
integer height = 88
integer taborder = 10
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
string text = "okay"
end type

event clicked;
boolean lb_com1=false,lb_com2=false
lb_com1=f_openport('COM1')
lb_com2=f_openport('COM3')

if lb_com1=false or lb_com2=false then
```

```
        messagebox('WEWS', 'Ccommunication Ports not Opened')

elseif lb_com1=true and lb_com2=true then
        messagebox('WEWS', 'Ccommunication Ports  Opened')
        per_em.enabled=true
        cb_2.enabled=true
end if



end event
type dw_1 from datawindow within main
integer x = 119
integer y = 164
integer width = 1056
integer height = 1192
string title = "none"
string dataobject = "dw_main"
boolean vscrollbar = true
boolean livescroll = true
borderstyle borderstyle = styleshadowbox!
end type

event rowfocuschanged;dw_1.SelectRow(0, FALSE)
dw_1.SelectRow(currentrow, TRUE)
end event

type gb_1 from groupbox within main
integer x = 1312
integer y = 208
integer width = 960
integer height = 240
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Open Communication Ports"
borderstyle borderstyle = stylelowered!
end type

type gb_2 from groupbox within main
integer x = 1312
```

```
integer y = 760
integer width = 960
integer height = 240
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Start wews"
borderstyle borderstyle = stylelowered!
end type

type gb_3 from groupbox within main
integer x = 1312
integer y = 464
integer width = 960
integer height = 240
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Period Of Scan"
borderstyle borderstyle = stylelowered!
end type

type gb_4 from groupbox within main
integer x = 1312
integer y = 1016
integer width = 960
integer height = 240
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
string text = "Exit"
borderstyle borderstyle = stylelowered!
```

```
end type

type gb_5 from groupbox within main
integer x = 1221
integer y = 164
integer width = 1129
integer height = 1192
integer textsize = -8
integer weight = 700
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
long backcolor = 67108864
borderstyle borderstyle = stylelowered!
end type


//
When wews system starting and running some errors may occur.Those are 300  Phone failure,128
Port Error,etc.These errors writen to wews_result.log file .The following source code developed
for user to trace errrors. //

global type w_log from window
end type
type mle_1 from multilineedit within w_log
end type
end forward

global type w_log from window
integer width = 2286
integer height = 1504
boolean titlebar = true
string title = "WEWS Log File --Mehmet Ugurlu"
boolean controlmenu = true
boolean minbox = true
boolean maxbox = true
boolean resizable = true
long backcolor = 67108864
mle_1 mle_1
end type
global w_log w_log

on w_log.create
this.mle_1=create mle_1
this.Control[]={this.mle_1}
```

```
end on

on w_log.destroy
destroy(this.mle_1)
end on

event open;//messagebox(Message.StringParm,string(mle_1.importfile(Message.StringParm)))


f_center(w_log)
integer li_FileNum

string ls_log_Input

long ll_FLength

ll_FLength = FileLength(LOG_FILE)

li_FileNum = FileOpen(LOG_FILE, StreamMode!)

IF ll_FLength < 32767 THEN
                FileRead(li_FileNum, ls_log_Input)

END IF

mle_1.text=ls_log_Input



end event
type mle_1 from multilineedit within w_log
integer x = 114
integer y = 224
integer width = 1952
integer height = 1028
integer textsize = -10
integer weight = 400
fontcharset fontcharset = ansi!
fontpitch fontpitch = variable!
fontfamily fontfamily = swiss!
string facename = "Arial"
long textcolor = 33554432
boolean hscrollbar = true
boolean vscrollbar = true
boolean displayonly = true
borderstyle borderstyle = StyleShadowBox!
end type
```

5.5 Wews User Interfaces

User interfaces improved by using a windows as or higher base envorment case tool is Power Builder 70, database orade 8i version.

User must enter all the data one by one. All thwe data is kept in system tables. The windows of wews system which are designed for the user are given in the following statements.

Figure 5.4 is an opening windows of wews system. Process menu of wews system is found in this windows. Process of Authentication and Authorization is done in this windows. After this process becomes successful, 'add ' and delete processes are carried out. As the program has just been loaded, there is no data. The user can enter a new data process by using add button on phone numbers tabpage. When add button is clicked, 'name', 'phone number', 'kind', 'valid' fields are enabled. After data is entered fields, the system does insertion into database. The deleting process of the saved row is by selecting the row on data-windows and clicking the delete button.After row deleted ,the system does delete current row from database permanently. The faults which occur while 'add and delete ' process is being carried out must be illustrated to the user as a message box.

As the program previously loaded, there is a data in system tables. After Authentication and Authorization process becomes successful, the datas which were inserted into the database before are retrieved. As it is stated above Add and Delete process is carried out. The faults which occur in the process of add and delete must be illustrated to the user as massage box.

Explanation of fields using in this window :

User ID: User identification when connect database

Password :User password

Source name: Database sourcename

Name: The name of person to be sending SMS

Phone No: Phone numbers of person to be sending SMS

Kind: Determines SMS kind which is going to be

Valid :Determines whether the data is valid or not in a wews system.

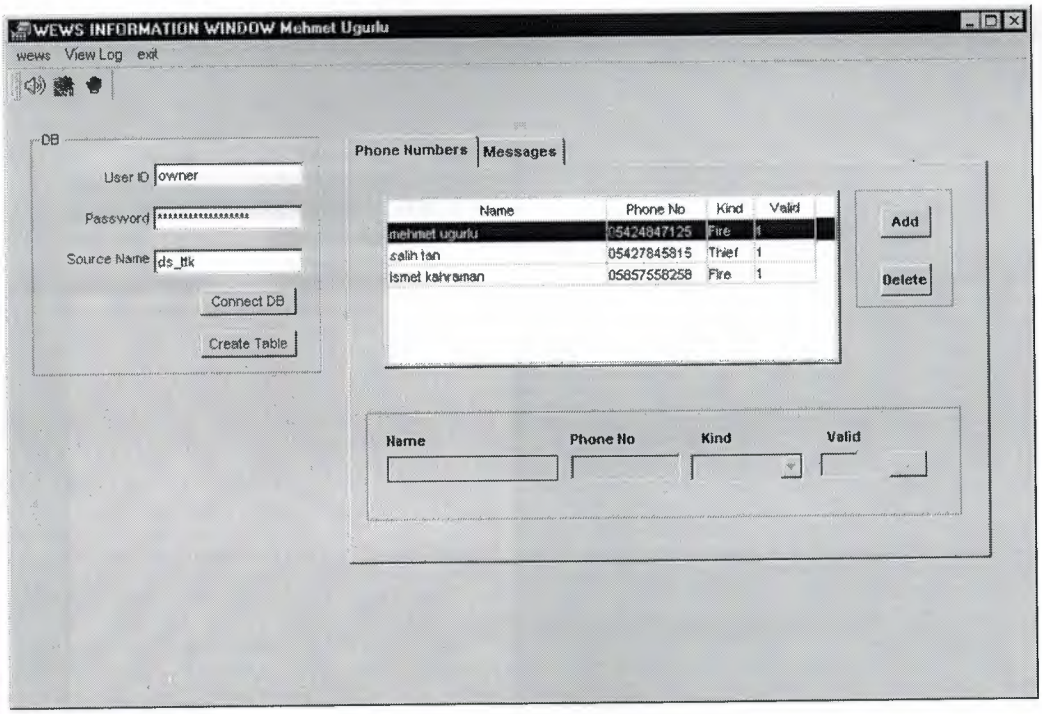


Figure 5.4 Opening windows of wews system(Main window)

Figure 5.5 is a windows which is designed for the wews system to operate. This windows is opened from the main menu wews section which is found on the menu. When the window is opened, the datas which were entered before are retrieved to urgent phone Datawindow.User opened port communication by clicking the 'okay' button, which founded in Open Port Communication section. When port communication becomes successfull,Period of Scan and 'okay' button ,which,founded in Start Wews section, is enabled. User entered period of scan value that is used for scaning the ports which sensors connected to.This field's edit mask is number.After that the wews system is started by clicking the ' okay ' button which is found in the start wews. When wews starting process becomes successful, the progress bar which is found below the window starts to operate. When wews starting process faills it gives a message and progress does not operate. If the process of parts to communication is an not successfull, it gives a massage and writes the error on the 'wews_ result.log file '. In order to exit from the wews system. It clicks 'exit ' button,which founded in exit section.

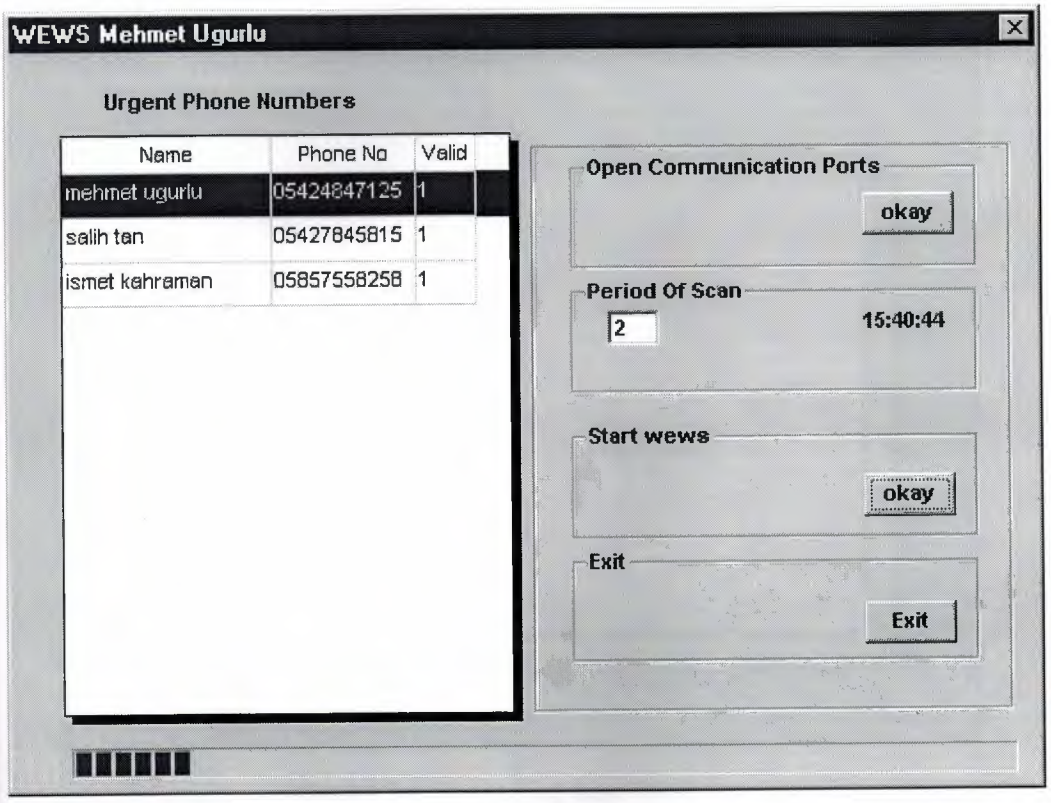


Figure 5.5 Wews Window

Figure 5.6 is designed to trace the errors which occur when wews system is operating. This window is opened from main menu wews log section which is show in Figure 5.4 menu.The errors which occur while wews systemis operating are written on wews-result.log file. These include the errors such as:

128 - Error

131 - Successful

100 - Port Delay Error

300 Phone Failure

330 Smsc (Message Service Center) Address Unknown

500 Unknown Error

Errors are written in the form of Error code error explanation the date and time when the error is occured.

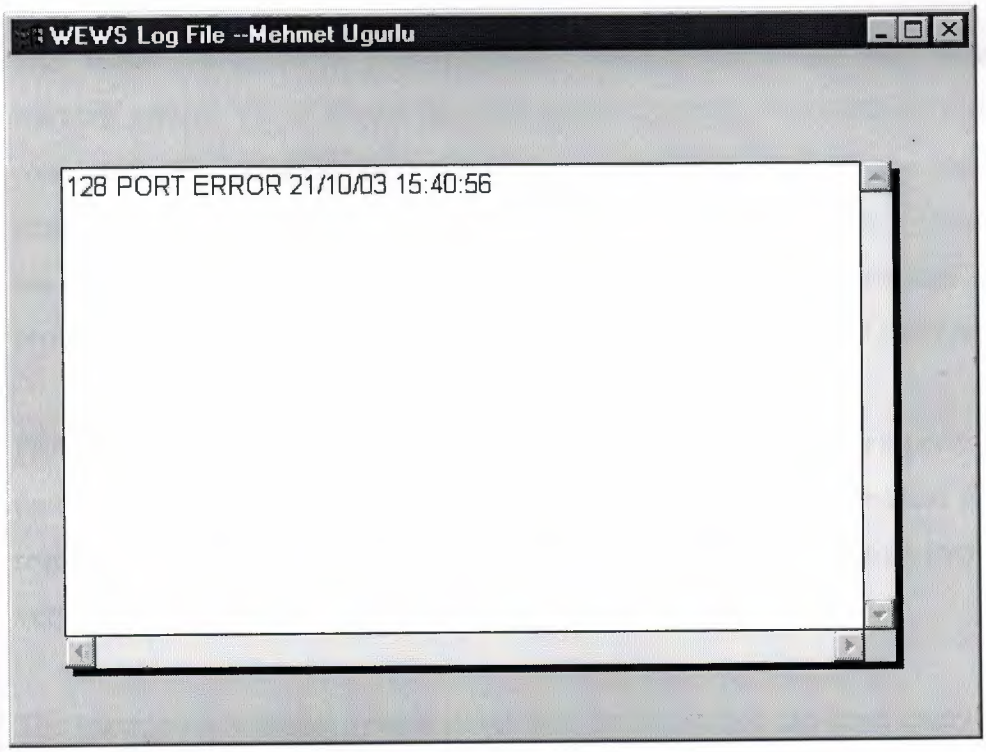


Figure 5.6 Wews Log File Window

Wews program is closed from the exit section which is found in main window menu in Figure 5.4

CONCLUSION

This thesis has described the design and development of an SMS based emergency and warning system. It is shown that the system hardware is based on a number of sensors, connected to a standard PC. The system receives data from the sensors and when an emergency is detected it sends SMS messages to pre-assigned mobile telephones. A software has been developed to transmit the messages using the SMS technology. The software also provides a log file so that the status of the message transfers can be checked.

Problems arised in this study may be enumerated this way: not properly initialize hardware ports, not able to read data from the ports, operating system dependent problems like losing responsiveness, not able to send SMS due to the problems originating from gsm operator, and very seldom not able to retrieve the data from database properly.

The emergency warning system developed by the author can have many applications in the commercial and industrial world.

This study can be used in emergency situations and businesses after a customization work done according to the fields' requirements.Possible application areas are discussed below:

Against car burglary, if sensors are located in the appropriate places of a car, it will automatically send warning messages to the appropriate receiver such as police, car owner.

In Gardening and green housing, observations made by means of appropriate sensors are sent via this system to the owner.

In highway public transportation, system can be integrated with tachographes and it will send warning messages to police and company owner in case driver exceeds the legal speed limits. And also in case of an accident, hospital personnel and other related parties can be alerted. Furthermore, if Global Positioning System is integrated in these vehicles, accident locations may also be appropriately informed.

In gas stations, amount of the gas can be informed to the company main data processing system transparently so that company may replenish the station's tanks. In freezing systems, sensible changes in the temperature can be instantly informed to the responsibles. In reservation required businesses, customers can be warned before the rendezvous. In health care systems, patients' situationes like heart beat, blood tension, body temperature and other related events can be informed to the doctors, patient relatives. In home security systems, in case of burglary, gas leakage, and fire, home owner and fire brigade can be warned.

Besides:

- Electricity, water, natural gas distribution systems, industry
- Smart buildings
- Villas
- Shopping centers
- Houses, hotels, automobile industry
- Banks

... and other possible areas seem eligible for this system application targets.

At above areas important processes occur other than alarms and crashes. These are applying periodical test scenarios at fire and security systems, providing security at the energy distributor center, operating feedback systems like generator. It provides necessary profits when necessary person become aware of the status of these kind of processes.

Various methods are used so as to inform the necessary places/person with true data. However most of the time human does the informing process. We must improve the system by automation and remove the human effect. For secure and safe delivery of the necessary information of critical alarms, important events, summary reports (stock, product, user, organization information etc..) to the authorized person, organizations prefers WEWS system. WEWS system evaluate the signals that it takes from the automation system or organization and transfer the necessary information to the predefined mobile telephone number by making use of SMS.

REFERENCES:

1. **Mobile Messaging Technologies and Services: SMS, EMS and MMS**
   Gwenael Le Bodic, January 2003
2. **The GSM Evolution: Mobile Packet Data Services**
   Peter Stuckmann, October 2002
3. **The Essential Guide to RF and Wireless**
   Carl J. Weisman, January 2002
4. **The Essential Guide to RF and Wireless**
   Carl J. Weisman, January 2000
5. **3G Wireless Networks**
   Clint Smith, September 2001
6. **Mobile Application Development with SMS and the SIM Toolkit**
   Scott C. Guthery, Mary Cronin, November 2001
7. **Complete Wireless Design**
   Cotter W. Sayre, January 2001
8. **Sensor Technologies and Data Requirements for ITS Applications**
   Lawrence A. Klein, June 2001
9. **Microsensors,MEMS Smart Devices**
   Julian W. Gardner, J. W. Gardner, Vijay K. Varadan, December 2001
10. **Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering**
    Joseph Mitola, September 2000
11. **Serial Port Complete: Programming and Circuits for RS-232 and RS-485 Links and Networks with Disk**
    Janet Louise Axelson, Jan Axelson, February 1999
12. **Programming the Parallel Port**
    Dhananjay V. Gadre, Dhananjay V. Garde, January 1998
13. **Electromechanical Sensors and Actuators**
    Ilene J. Busch-Vishniac, I. Busch-Vishniac, October 1998

14. **Fundamentals of Programmable Logic Controllers, Sensors, and Communications**

Jon Stenerson, June 1998

15. **GSM and Personal Communications Handbook**

Siegmund Redl, Matthias Weber, Malcolm Oliphant, Malcolm W. Oliphant, May 1998

16. **Multi-Sensor Fusion : Fundamentals and Applications with Software**

Richard R. Brooks, Sundararaja Iyengar, S. S. Iyengar, November 1997

17. **Parallel Port Complete**

Janet Louise Axelson, Jan Axelson, March 1997

18. **Parallel Programming Using C++**

Gregory V. Wilson, Paul Lu (Editor), Foreword by Bjane Stroustrup, August 1996

19. **Mobile Radio Communications**

Raymond Steele, January 1996

20. **Acoustic Wave Sensors: Theory, Design, Physico-Chemical Applications**

David Stephen Ballantine, D. S. Ballantine, Greg C. Frye, December 1995

21. **Introduction to GSM**

Siegmund Redl, Malcolm W. Oliphant, Mathias K. Weber, Mathias Weber, Matthias Weber, March 1995

22. **Sensors and Control Systems in Manufacturing**

Sabrie Soloman,1994

23. **Expert C Programming**

Peter van der Linden, P. Van Der Linden, June 1994

24. **C Programming Language**

Brian W. Kernighan, Dennis M. Ritchie, Dennis Ritchie, March 1989

25. **Digital Cellular Mobile Radio Links And Networks**

Y:F:Ko,December 1989

26. **Electrochemical Sensors in Immunological Analysis**

T. Ngo, June 1987

27. Advanced Mobile PHOne services:The Cellular Concept

J.Vol,january 1979

28. ETSI(European Telecommunication Standarts Institue) ,

http://www.etsi.com/

29. AT Command Set For Nokia GSM Products ,

http://3ton.com/besik/ATNOKIA.pdf

30.Turkcell Co.Turkey

http://www.turkcell.com.tr

31.Telsim Co.Turkey

http://www.telsim.com.tr

32. http://www.rtcu.dk/OnlineHelp/examples_-_greenhouse-2_(advanced).htm

33. http://www.wilke-technology.com/html/kundenref_deboosere.html

34. http://www.pq-asia.com/PMmsgmaster.htm

35. http://www.processingtalk.com/news/pbe/pbe100.html

36.http://www.siemens.com

APPENDIX 1

Green House

Green House project,which was developed in Denmark by Logic IO Corporation under the unit of Remote Telemetry and Control Units.In this project; Gardener Green has a greenhouse in which he wants to control the temperature .For that purpose he has bought a couple of temprature sensors and some actuators.One of the temperature sensors will be adjusted to give a signal when the temperature falls below a given temperature, and the other sensor will give a signal when the temperature rises above a given temprature.Using inputs from these sensores he wants to turn on a heating unit and turn on the ventilation duct. If the system does not succeed in bringing the temperature within the given limits,the gardener wants to have a SMS message sent to his mobile telephone. This message will notify him about the status of the green house.

(Ref: **32. http://www.rtcu.dk/OnlineHelp/examples_-_greenhouse-2_(advanced).htm**)

GSM Alarm Engine

GSM Alarm Engine project ,which was developed by Deboosere Telecom in Belgium.

Designed for remote monitoring, command and control of machines 16 digital inputs 16 digital outputs (5V - 20mA)Optional extension-board 16 relais outputs (1A - 250Vac).Automatic transmission of alarms via SMS to one or multiple persons.

D16 defineable SMS-messages of 160 characters each.Sabotage-proof against unauthorized access.Dual band GSM-modem 900/1800Mhz.
Configuration-software. Soft- and hardware development DEBOOSERE TELECOMAutonomous detection of level (sensor) in glascontainer. The container sends an SMS-message when it is 75% full (level adaptable). When the container is not emptied within 8 hours after the message "FULL" then it re-sends an alarm to the central software. When the container is emptied, he will send an acknowledgement .

"EMPTY", so the dispatcher knows everything is OK.Low-power mode because of the lack of electricity.Internal battery must function 3 years (recharging by solar panel) – possibility to check battery-status via SMS.Changing of parameters (time to report status, answer-number, automatic answer, test-message) via SMS. Figure App-1.1

(Ref: **33. http://www.wilke-technology.com/html/kundenref_deboosere.html**)

Figure App-1.1 GSM Alarm Engine project

Palm Size Alarm Monitering System

Palm Size Alarm Monitering System was developed in Singapora by PQ-Asia Group

General Features:

a. A Complete Monitoring System. No PC is required to operate this product.

b. Send SMS alarm message up to 40 mobile handset numbers. Able to assign alarm to different recipient by phone number grouping.

c. User friendly. Web based interface using common browser software. E.g. Internet Explorer or Netscape.

d. No programming knowledge required to setup the system. Just click and save.

e. Ethernet ready. Networkable on any existing Local Area Network (LAN).

e. Award winning product in Singapore Industrial Automation Award 2003.

(Ref: **34. http://www.pq-asia.com/PMmsgmaster.htm**)

The System Ceres modular

The System Ceres modular ,which was developed in United Kingdom by PBE System Corporation. The System Ceres modular control solution from PBE Systems harvests the power of the Internet with GSM/SMS messages to provide remote monitoring and control functions of manufacturing plant, alarms and equipment directly to your mobile phone using SMS text messages, Internet, or PC.

This versatile and powerful system comprises two stand-alone components. The "Janus" communication model can simply be used on its own as a sophisticated logic controller, alarm and data logging device, or it can be combined with " DaJaView", a web-enabled SQL database which receives information and alarms from field-based Janus units in SMS or CSV format displaying the results on a website viewable over any internet connection.

Janus modules provide bi-directional communications and control functions on up to 200 telephone numbers via SMS messages, Internet, or PC.

All functions can be accessed, programmed and controlled by any GSM mobile phone, data logging and downloading information to a schedule or demand. Multiple units can be linked to create a network.Alarms with many levels of logic can be sent to multiple numbers using SMS messages, which themselves may be changed by any mobile phone.Current and historical data of the target device can be viewed via GSM, Internet or PC.When viewed by PC the data can be manipulated by standard windows based software.The DaJaView SQL database is hosted on a central server, displaying information sent by Janus field units and is viewable through any standard web browser.Information can be manipulated in different user selected formats and access may be limited by security control.DaJaView can also send e-mails and SMS messages on alarm activation.

(Ref: **35. http://www.processingtalk.com/news/pbe/pbe100.html**)

GSM Engine TC35

GSM Engine TC35 which was developed in Germany by Siemens Corporation.
In figure App-1.2 show to genaral configuration of GSM Engine TC35
The reference configuration, consisting of

GSM Engine
Development Support Box
Sim Card reader integrated on the DSB
Handset
Pc

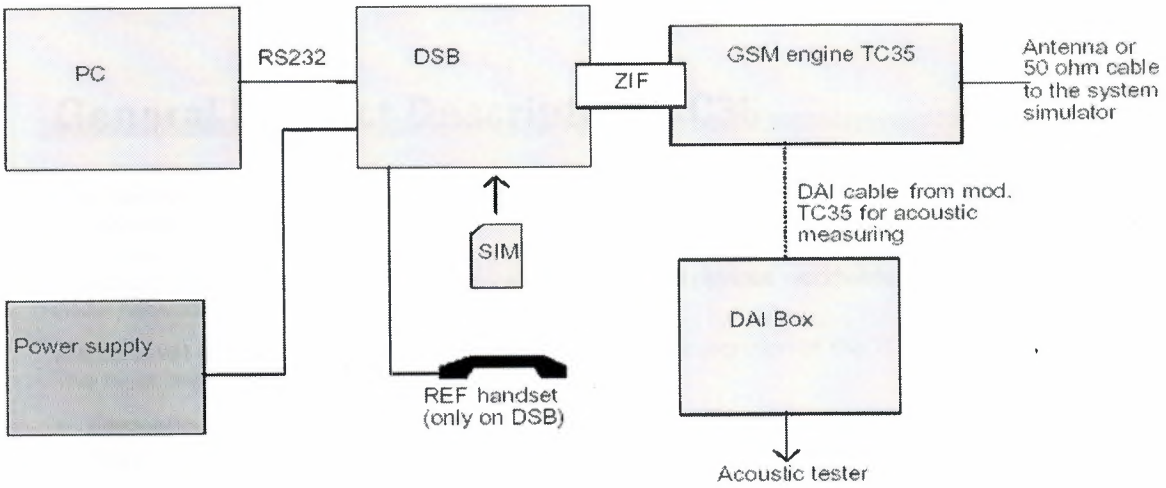


Figure App-1.2 Genaral configuration of GSM Engine TC35

(Ref: **36.www.siemens.com**)

General Product Description TC35

The Siemens TC35 represents the new generation of GSM modules in dual band technology EGSM900/GSM1800. TC35 is lightweight and small, has optimal power consumption and transmits data, voice, SMS and fax. The TC35 is suitable for complex industrial applications such as telemetry, telematics or communication, POS terminals and handheld devices worldwide wherever there is a GSM network.

Siemens will offer a corresponding evaluation kit for rapid integration of the TC35.
The most important features are as follows:

- Frequency EGSM900/GSM1800 Phase 2+
- GSM Class: Small MS
- 2W – Power for Class 4/ EGSM900
- 1W – Power for Class 1/ GSM1800
- Voice transmission with Triple Rate Codec (HR, FR, EFR)
- Data transmission in CSD up to 9.6 kbps in non-transparent mode and V110
- Analog audio interface
- Standard handsfree function
- FAX
- SMS (text and PDU mode) /MT/MO/CB
- Software download possible via RS232 and SIM card reader
- BOF with AT operations set compatible with M20/A20 with the reference status: M20 Technical Description Version 7.0 and M20 SW Version 3.3 as well as A20 Technical Description Version 1.0
- Electrical interface via 40-pin ZIF connector (AVX 04-6240 or similar structure)
- HF connector: coax jack 50Ω Murata GSC, MM9329-2700
- SIM card reader interface 3V via 40-pin ZIF connector
- Reset using ATC or PD (Power Down) via line at the ZIF connector
- RS 232 autobauding (4.8 kbps – 115 kbps)
- Power ON using ignition line at the ZIF connector
- Power OFF using PD (Power Down) via line at the ZIF connector or ATC
- Locating and position services possible with AT^MONI, AT^MONP
- Temperature range from –20°C to +55°C
- Supply voltage from 3.3V to 5.5V
- Power consumption idle mode < 3.5mA
- Power consumption active mode ~ 300mA
- Power consumption peak 1.6 – 2.3A
- Power consumption in the OFF state ~ 100µA
- Dimensions (L,B,H) 54.5x36x6.7 mm
- Volume < 13 cm^3
- Weight ~ 18 g

APPENDIX 2

GSM AT-Commands for SMS

This AT-Commands related to ETSI (European Telecommunications Standards Institute)

5.1.1 AT+CMGC Send an SMS command	
Test command AT+CMGC=?	Response OK

Write command
 	
if text mode (AT+CMGF=1):
AT+CMGC=<fo>,<ct>[,<pid>[,<mn>[,<da>[,<toda>]]]]<CR>
text is entered
<ctrl-Z/ESC>
	Response
	
	if text mode (+CMGF=1) and sending successful:
	+CMGC: <mr>[,<scts>]
	if sending fails:
	+CMS ERROR: <err>

if PDU mode (AT+CMGF=0):
AT+CMGC=<length><CR>
PDU is given
<ctrl-Z/ESC>
+CMGC=?
	Response
	
	if PDU mode (+CMGF=0) and sending successful:
	+CMGC: <mr>[,<ackpdu>]
	if sending fails:
	+CMS ERROR: <err>
	
	Parameter
	<length> Length of PDU
	<pdu> See "AT+CMGL"
	<mr> Message reference
	<fo> depending on the command or result code: first octet of GSM 03.40 SMS-DELIVER, SMS-SUBMIT (default 17), SMS-STATUS-REPORT, or SMS-COMMAND (default 2) in integer format
	<ct> GSM 03.40 TP-Command-Type in integer format (default 0)
	<pid> GSM 03.40 TP-Protocol-Identifier in integer format (default 0)
	<toda> GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of <da> is + (IRA 43) default is 145, otherwise default is 129)
	<da> GSM 03.40 TP-Destination-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by **<toda>**

Reference GSM 07.05	Note

5.1.2 AT+CMGD Delete SMS message

Test command AT+CMGD=?	Response **OK** Parameter
Execute command AT+CMGD= <index>	Response TA deletes message from preferred message storage <**mem1**> location <**index**>. **OK** If error is related to ME functionality: **+CMS ERROR <err>** Parameter <**index**> integer type; value in the range of location numbers supported by the associated memory
Reference GSM 07.05	Note

5.1.3 AT+CMGF Select SMS message format

Test command AT+CMGF=?	Response **+CMGF:** (list of supported <**mode**>s) **OK** Parameter see set command
Read command AT+CMGF?	Response **+CMGF: <mode> OK** Parameter see set command
Set command AT+CMGF = [<mode>]	Response TA sets parameter which specifies the input and output format of messages to be used. **OK** Parameter <**mode**> 0 PDU mode 1 text mode
Reference GSM 07.05	Note

5.1.4 AT+CMGL List SMS messages from preferred store

Test command AT+CMGL=?	Response +CMGL: (list of supported <stat>s) OK Parameter See execute command
Execute command AT+CMGL [=<stat>]	Parameter **1) If text mode:** <stat> "REC UNREAD" Received unread messages (default) "REC READ" Received read messages "STO UNSENT" Stored unsent messages "STO SENT" Stored sent messages "ALL" All messages **2) If PDU mode:** <stat> 0 Received unread messages (default) 1 Received read messages 2 Stored unsent messages 3 Stored sent messages 4 All messages Response TA returns messages with status value <stat> from message storage <mem1> to the TE. If status of the message is 'received unread', status in the storage changes to 'received read'. *Note:* if the selected <mem1> can contain different types of SMs (e.g. SMS-DELIVERs, SMS-SUBMITs, SMS-STATUS-REPORTs and SMS-COMMANDs), the response may be a mix of the responses of different SM types. TE application can recognize the response format by examining the third response parameter.
	Response 1) If text mode (+CMGF=1) and command successful: for SMS-SUBMITs and/or SMS-DELIVERs: **+CMGL: <index>,<stat>,<oa/da>,[<alpha>],[<scts>][,<tooa/toda>,** **<length>]<CR><LF><data>[<CR><LF>** **+CMGL: <index>,<stat>,<da/oa>,[<alpha>],[<scts>][,<tooa/toda>,** **<length>]<CR><LF><data>[...]] OK** for SMS-STATUS-REPORTs: **+CMGL: <index>,<stat>,<fo>,<mr>,[<ra>],[<tora>],<scts>,<dt>,<st>** **[<CR><LF>** **+CMGL: <index>,<stat>,<fo>,<mr>,[<ra>],[<tora>],<scts>,<dt>,<st>** **[...]] OK** for SMS-COMMANDs: +CMGL: <index>,<stat>,<fo>,<ct>[<CR><LF>** +CMGL: <index>,<stat>,<fo>,<ct>[...]] OK** for CBM storage: +CMGL: <index>,<stat>,<sn>,<mid>,<page>,<pages>** <CR><LF><data>[<CR><LF>** +CMGL: <index>,<stat>,<sn>,<mid>,<page>,<pages>** <CR><LF><data>[...[]OK** **2) If PDU mode (+CMGF=0) and command successful:** +CMGL: <index>,<stat>,[<alpha>],<length><CR><LF><pdu>** [<CR><LF>+CMGL: <index>,<stat>,[alpha],<length><CR><LF><pdu>**

	[...]] OK 3) If error is related to ME functionality: +CMS ERROR: <err>
	Parameter <alpha> string type alphanumeric representation of <da> or <oa> corresponding to the entry found in MT phonebook; implementation of this feature is manufacturer- specific <ct> GSM 03.40 TP-Command-Type in integer format (default 0) <da> GSM 03.40 TP-Destination-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by <toda> <data> In the case of SMS: GSM 03.40 TP-User-Data in text mode responses; format: -if <dcs> indicates that GSM 03.38 default alphabet is used and <fo> indicates that GSM 03.40 TP-User-Data-Header-Indication is not set: ME/TA converts GSM alphabet into current TE character set according to rules of Annex A -if <dcs> indicates that 8-bit or UCS2 data coding scheme is used, or <fo> indicates that GSM 03.40 TP-User-Data-Header-Indication is set: ME/TA converts each 8-bit octet into hexadecimal numbers containing two IRA characters (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)) In the case of CBS: GSM 03.41 CBM Content of Message in text mode responses; format: - if <dcs> indicates that GSM 03.38 default alphabet is used: ME/TA converts GSM alphabet into current TE character set according to rules of Annex A -if <dcs> indicates that 8-bit or UCS2 data coding scheme is used: ME/TA converts each 8-bit octet into hexadecimal numbers containing two IRA characters
	Parameter <dt> GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd,hh:mm:ss±zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08" <fo> depending on the command or result code: first octet of GSM 03.40 SMS-DELIVER, SMS-SUBMIT (default 17), SMS-STATUS-REPORT, or SMS-COMMAND (default 2) in integer format <length> integer type value indicating in the text mode (+CMGF=1) the length of the message body <data> (or <cdata>) in characters; or in PDU mode (+CMGF=0), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length) <index> integer type; value in the range of location numbers supported by the associated memory <mid> GSM 03.41 CBM Message Identifier in integer format <mr> GSM 03.40 TP-Message-Reference in integer format <oa> GSM 03.40 TP-Originating-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by <tooa> <pages> GSM 03.41 CBM Page Parameter bits 0-3 in integer format <pdu> In the case of SMS: GSM 04.11 SC address followed by GSM 03.40 TPDU in hexadecimal format: ME/TA converts each octet of TP data unit into hexadecimal numbers containing two IRA characters (e.g. octet with

		integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)). In the case of CBS: GSM 03.41 TPDU in hexadecimal format.
	<page>	GSM 03.41 CBM Page Parameter bits 4-7 in integer format
	<ra>	GSM 03.40 TP-Recipient-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by <tora>
	<scts>	GSM 03.40 TP-Service-Centre-Time-Stamp in time-string format (refer <dt>)
	<sn>	GSM 03.41 CBM Serial Number in integer format
	<st>	GSM 03.40 TP-Status in integer format
	<toda>	GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of <da> is + (IRA 43) default is 145, otherwise default is 129)
	<tooa>	GSM 04.11 TP-Originating-Address Type-of-Address octet in integer format (default refer<toda>)
	<tora>	GSM 04.11 TP-Recipient-Address Type-of-Address octet in integer format (default refer<toda>)
Reference GSM 07.05	Note	

5.1.5 AT+CMGR Read SMS message

Test command AT+CMGR=?	Response OK Parameter
Execute command AT+CMGR= <index>	Parameter <**index**> integer type; value in the range of location numbers supported by the associated memory Response TA returns SMS message with location value <index> from message storage <**mem1**> to the TE. If status of the message is 'received unread', status in the storage changes to 'received read'. 1) If text mode (**+CMGF=1**) and command successful: for SMS-DELIVER: **+CMGR: <stat>,<oa>,[<alpha>],<scts> [,<tooa>,<fo>,<pid>,<dcs>,<sca>,<tosca>,<length>]<CR><LF><data>** for SMS-SUBMIT: **+CMGR: <stat>,<da>,[<alpha>] [,<toda>,<fo>,<pid>,<dcs>,[<vp>],<sca>,<tosca>,<length>]<CR><LF><data>** for SMS-STATUS-REPORT: **+CMGR: <stat>,<fo>,<mr>,[<ra>[,]<tora>],<scts>,<dt>,<st>** for SMS-COMMAND: **+CMGR: <stat>,<fo>,<ct> [,<pid>,[<mn>],[<da>],[<toda>],<length><CR><LF><cdata>]** for CBM storage: **+CMGR: <stat>,<sn>,<mid>,<dcs>,<page>,<pages><CR><LF><data>** 2) If PDU mode (+CMGF=0) and command successful: **+CMGR: <stat>,[<alpha>],<length><CR><LF><pdu> OK** 3)If error is related to ME functionality: **+CMS ERROR: <err>** Parameter <**alpha**> string type alphanumeric representation of <da> or <oa> corresponding to the entry found in MT phonebook; implementation of this feature is manufacturer-specific <**stat**> integer type in PDU mode (default 0), or string type in text mode (default "REC UNREAD"); indicates the status of message in memory: defined values: 0 "REC UNREAD" received unread message (i.e. new message) 1 "REC READ" received read message 2 "STO UNSENT" stored unsent message (only applicable to SMs) 3 "STO SENT" stored sent message (only applicable to SMs) 4 "ALL" all messages (only applicable to AT+CMGL List SMS messages from preferred store command)

	<ct> GSM 03.40 TP-Command-Type in integer format (default 0) <da> GSM 03.40 TP-Destination-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by <toda> <data> In the case of SMS: GSM 03.40 TP-User-Data in text mode responses; format: -if <dcs> indicates that GSM 03.38 default alphabet is used and <fo> indicates that GSM 03.40 TP-User-Data-Header-Indication is not set: ME/TA converts GSM alphabet into current TE character set according to rules covered in Annex A -if <dcs> indicates that 8-bit or UCS2 data coding scheme is used, or <fo> indicates that GSM 03.40 TP-User-Data-Header-Indication is set: ME/TA converts each 8-bit octet into hexadecimal numbers containing two IRA characters (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)) In the case of CBS: GSM 03.41 CBM Content of Message in text mode responses; format:
	Parameter - if <dcs> indicates that GSM 03.38 default alphabet is used: ME/TA converts GSM alphabet into current TE character set according to rules covered in Annex A -if <dcs> indicates that 8-bit or UCS2 data coding scheme is used: ME/TA converts each 8-bit octet into hexadecimal numbers containing two IRA characters <dcs> depending on the command or result code: GSM 03.38 SMS Data Coding Scheme (default 0), or Cell Broadcast Data Coding Scheme in integer format <cdata> GSM 03.40 TP-Command-Data in text mode responses; ME/TA converts each 8-bit octet into two IRA character long hexadecimal number (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)) <dt> GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd,hh:mm:ss±zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08" <fo> depending on the command or result code: first octet of GSM 03.40 SMS- DELIVER, SMS-SUBMIT (default 17), SMS-STATUS-REPORT, or SMS-COMMAND (default 2) in integer format <length> integer type value indicating in text mode (+CMGF=1) the length of the message body <data> (or <cdata>) in characters; or in PDU mode (+CMGF=0), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length) <index> integer type; value in the range of location numbers supported by the associated memory <mid> GSM 03.41 CBM Message Identifier in integer format <mr> GSM 03.40 TP-Message-Reference in integer format <oa> GSM 03.40 TP-Originating-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by <tooa> <page> GSM 03.41 CBM Page Parameter bits 4-7 in integer format <pages> GSM 03.41 CBM Page Parameter bits 0-3 in integer format

	<pdu>	In the case of SMS: GSM 04.11 SC address followed by GSM 03.40 TPDU in hexadecimal format: ME/TA converts each octet of TP data unit into hexadecimal numbers containing two IRA characters (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)). In the case of CBS: <ra> GSM 03.40 TP-Recipient-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by <tora>
	<pid>	GSM 03.40 TP-Protocol-Identifier in integer format (default 0)
	<ra>	GSM 03.40 TP-Recipient-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted to characters of the currently selected TE character set (refer command AT+CSCS Select TE character set.); type of address given by <tora>
	<sca>	GSM 04.11 RP SC address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted to characters of the currently selected TE character set (refer command AT+CSCS Select TE character set.); type of address given by <tosca>
	<scts>	GSM 03.40 TP-Service-Centre-Time-Stamp in time-string format (refer <dt>)
	<sn>	GSM 03.41 CBM Serial Number in integer format
	<st>	GSM 03.40 TP-Status in integer format
	<toda>	GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of <da> is + (IRA 43) default is 145, otherwise default is 129)
	<tooa>	GSM 04.11 TP-Originating-Address Type-of-Address octet in integer format (default refer<toda>)
	<tora>	GSM 04.11 TP-Recipient-Address Type-of-Address octet in integer format (default refer<toda>)
	<tosca>	GSM 04.11 RP SC address Type-of-Address octet in integer format (default refer <toda>)
	<vp>	depending on SMS-SUBMIT <fo> setting: GSM 03.40 TP-Validity-Period either in integer format (default 167) or in time-string format (refer <dt>)
Reference GSM 07.05	Note	

5.1.6 AT+CMGS Send SMS message

Test command AT+CMGS=?	Response **OK** Parameter
Execute command 1) If text mode (+CMGF=1): +CMGS=<da>[,<toda>]<CR> text is entered <ctrl-Z/ESC> 2) If PDU mode (+CMGF=0): +CMGS=<length><CR> PDU is given <ctrl-Z/ESC> ESC aborts message	Response TA transmits SMS message from a TE to the network (SMS-SUBMIT). Message reference value <mr> is returned to the TE on successful message delivery. Value can be used to identify message upon unsolicited delivery status report result code. 1) If text mode (+CMGF=1) and sending successful: **+CMGS: <mr>[,scts>] OK** 2) If PDU mode (+CMGF=0) and sending successful: **+CMGS: <mr>[,ackpdu>] OK** 3) If error is related to ME functionality: **+CMS ERROR: <err>**

Parameter

<da>	GSM 03.40 TP-Destination-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by <toda>
<toda>	GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of <da> is + (IRA 43) default is 145, otherwise default is 129)
<length>	integer type value indicating in the text mode (+CMGF=1) the length of the message body <data> (or <cdata>) in characters; or in PDU mode (+CMGF=0), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length)
<mr>	GSM 03.40 TP-Message-Reference in integer format
<scts>	GSM 03.40 TP-Service-Centre-Time-Stamp in time-string format (refer <dt>)
<dt>	GSM 03.40 TP-Discharge-Time in time-string format: "yy/MM/dd,hh:mm:ss±zz", where characters indicate year (two last digits), month, day, hour, minutes, seconds and time zone. For example, 6th of May 1994, 22:10:00 GMT+2 hours equals "94/05/06,22:10:00+08"
<ackpdu>	GSM 03.40 RP-User-Data element of RP-ACK PDU; format is same as for **<pdu>** in case of SMS, but without GSM 04.11 SC address field and parameter shall be enclosed in double quote characters like a normal string type parameter
<pdu>	In the case of SMS: GSM 04.11 SC address followed by GSM 03.40 TPDU in hexadecimal format: ME/TA converts each octet of TP data unit into hexadecimal numbers containing two IRA characters (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)). In the case of CBS: GSM 03.41 TPDU in hexadecimal format.

5.1.7 AT+CMGW Write SMS message to memory

Test command AT+CMGW=?	Response OK Parameter
Execute command 1) If text mode (+CMGF=1): +CMGW[=<oa/da>[,<tooa/toda>[,stat>]]] <CR> text is entered ctrl-Z/ESC> <ESC> quits without sending 2) If PDU mode (+CMGF=0): +CMGW=<length>[,stat]<CR> PDU is given <ctrl-Z/ESC>	Response TA transmits SMS message (either SMS-DELIVER or SMS-SUBMIT) from TE to memory storage **<mem2>**. Memory location **<index>** of the stored message is returned. Message status will be set to 'stored unsent' unless otherwise given in parameter <stat>. *Note:* SMS-COMMANDs and SMS-STATUS-REPORTs can not be stored in text mode. If writing is successful: **+CMGW: <index> OK** If error is related to ME functionality: **+CMS ERROR: <err>** Parameter **<oa>** GSM 03.40 TP-Originating-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by **<tooa>** **<da>** GSM 03.40 TP-Destination-Address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by **<toda>** **<tooa>** GSM 04.11 TP-Originating-Address Type-of-Address octet in integer format (default refer **<toda>**) **<toda>** GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of **<da>** is + (IRA 43) default is 145, otherwise default is 129) **<length>** integer type value indicating in the text mode (**+CMGF=1**) the length of the message body **<data>** (or **<cdata>**) in characters; or in PDU mode (**+CMGF=0**), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length) **<stat>** 0 "REC UNREAD" Received unread messages (default) 1 "REC READ" Received read messages 2 "STO UNSENT" Stored unsent messages 3 "STO SENT" Stored sent messages 4 "ALL" All messages **<pdu>** In the case of SMS: GSM 04.11 SC address followed by GSM 03.40 TPDU in hexadecimal format: ME/TA converts each octet of TP data unit into hexadecimal numbers containing two IRA characters (e.g. octet with integer value 42 is presented to TE as two characters 2A (IRA 50 and 65)). In the case of CBS: GSM 03.41 TPDU in hexadecimal format. **<index>** Index of message in selected storage **<mem2>** *Note:* ctrl-Z sends/writes message, Returns **Ok** ESC aborts input, message NOT sent/written. Returns **Ok**

5.1.8 AT+CMSS Send SMS message from storage

Test command AT+CMSS=?	Response **OK** Parameter
Execute command +CMSS= <index>[,<da>[,<toda>]]	Response TA sends message with location value **<index>** from message storage **<mem2>** to the network (SMS-SUBMIT or SMS-COMMAND). If new recipient address **<da>** is given for SMS-SUBMIT, it shall be used instead of the one stored with the message. Reference value **<mr>** is returned to the TE on successful message delivery. Values can be used to identify message upon unsolicited delivery status report result code. This command should be abortable. 1) If text mode (**+CMGF=1**) and send successful: **+CMSS: <mr>[,scts>] OK** 2) If PDU mode (**+CMGF=0**) and send successful: **+CMSS: <mr>[,ackpdu>] OK** 3) If error is related to ME functionality: **+CMS ERROR: <err>** Parameter **<ackpdu>** GSM 03.40 RP-User-Data element of RP-ACK PDU; format is same as for <pdu> in case of SMS, but without GSM 04.11 SC address field and parameter shall be bounded by double quote characters like a normal string type parameter. **<index>** integer type; value in the range of location numbers supported by the associated memory **<da>** GSM 03.40 TP-Destination-Address Address-Value field in'string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by **<toda>** **<scts>** GSM 03.40 TP-Service-Centre-Time-Stamp in time-string format. **<toda>** GSM 04.11 TP-Destination-Address Type-of-Address octet in integer format (when first character of **<da>** is + (IRA 43) default is 145, otherwise default is 129) **<mr>** GSM 03.40 TP-Message-Reference in integer format

5.1.9 AT+CNMA New SMS message acknowledge to ME/TE, only phase 2+

Test command AT+CNMA=?	Response 1) If text mode (+CMGF=1): **OK** 2) If PDU mode (+CMGF=0): **+CNMA:** (list of supported **<n>**s) **OK** Parameters see execute command
Execute command 1) If text mode: AT+CNMA 2) If PDU mode: AT+CN-MA[=<n>[,<length>[<CR>**PDU is given**<ctrl-Z/ESC>]]]	Response TA confirms successful receipt of a new message (SMS-DELIVER or SMS-STATUS-REPORT) which is routed directly to the TE. TA shall not send another +CMT or +CDS result code to TE until previous one is acknowledged. If ME does not receive acknowledgment within required time (network timeout), ME should send RP-ERROR to the network. TA shall automatically disable routing to TE by setting both **<mt>** and **<ds>** values of +CNMI to zero. *Note:* the command shall only be used when +CSMS parameter <service> equals 1 (= phase 2+). 1) If text mode: **OK** 2) If PDU mode: **OK** 3) If error is related to ME functionality: **+CMS ERROR:** <err> Parameters **<n>** 0 command operates similarly as defined for the text mode 1 send RP-ACK (or buffered result code received correctly) 2 send RP-ERROR (if PDU is not given, ME/TA shall send SMS-DELIVER-REPORT with GSM 03.40 TP-FCS value set to 'FF' (unspecified error cause)) **<length>** integer type value indicating in the text mode (+CMGF=1) the length of the message body **<data>** (or <cdata>) in characters; or in PDU mode (+CMGF=0), the length of the actual TP data unit in octets (i.e. the RP layer SMSC address octets are not counted in the length)

5.1.10 AT+CNMI New SMS message indications

Test command AT+CNMI=?	Response **+CNMI:** (list of supported **<mode>s**), (list of supported **<mt>s**), (list of supported **<bm>s**), (list of supported **<ds>s**), (list of supported **<bfr>s**) **OK** Parameter see set command
Read command AT+CNMI?	Response **+CNMI: <mode>,<mt>,<bm>,<ds>,<bfr> OK** Parameter see set command
Set command AT+CNMI = [<mode> [,<mt>[,<bm> [,<ds>[,<bfr>]]]]]	Response TA selects the procedure, how the receipt of new SMS messages from the network is indicated to the TE when TE is active, e.g. DTR signal is ON. If TE is inactive (e.g. DTR signal is OFF), message receiving should be done as specified in GSM 03.38. *Note1:* when DTR signal is not available or the state of the signal is ignored (V.25ter command &D0), reliable message transfer can be assured by using **+CNMA** acknowledgment procedure. *Note2:* the rules **<mt>=2** and **<mt>=3** for storing received SM are possible *only if phase 2+* compatibility is activated with **+CSMS=1** **OK** If error is related to ME functionality: **+CMS ERROR: <err>** Parameter **<mode>** 0 Buffer unsolicited result codes in the TA. If TA result code buffer is full, indications can be buffered in some other place or the oldest indications may be discarded and replaced with the new received indications. 1 Discard indication and reject new received message unsolicited result codes when TA-TE link is reserved (e.g. in on-line data mode). Otherwise forward them directly to the TE. 2 Buffer unsolicited result codes in the TA when TA-TE link is reserved (e.g. in on-line data mode) and flush them to the TE after reservation. Otherwise forward them directly to the TE. 3 Forward unsolicited result codes directly to the TE. TA-TE link specific inband technique used to embed result codes and data when TA is in on-line data mode. **<mt>** (the rules for storing received SMs depend on the relevant data coding method (refer to GSM 03.38 [2]), preferred memory storage (+CPMS) setting and this value *Note*: if AT command interface is acting as the only display device, the ME must support storage of class 0 messages and messages in the message waiting indication group (discard message) 0 No SMS-DELIVER indications are routed to the TE. 1 If SMS-DELIVER is stored in ME/TA, indication of the memory location is routed to the TE using unsolicited result code: **+CMTI: <mem>,<index>** 2 SMS-DELIVERs (except class 2 messages and messages in the message waiting indication group (store message)) are routed directly to the TE using unsolicited result code: +CMT: [<alpha>], <length><CR><LF><pdu> (PDU mode enabled) 3 Class 3 SMS-DELIVERs are routed directly to the TE using unsolicited result codes defined in <mt>=2. Messages of other data coding schemes result in indication as defined in <mt>=1.

	<bm> (the rules for storing received CBMs depend on the relevant data coding method (refer to GSM 03.38 [2]), the setting of Select CBM Types (+CSCB) and this value: 0 No CBM indications are routed to the TE. 1 If CBM is stored in ME/TA, indication of the memory location is routed to the TE using unsolicited result code: **+CBMI: <mem>,<index>** 2 New CBMs are routed directly to the TE using unsolicited result code: **+CBM: <length><CR><LF><pdu>** (PDU mode enabled) or **+CBM: <sn>,<mid>,<dcs>,<page>,<pages><CR><LF><data>** (text mode enabled) If ME supports data coding groups which define special routing also for messages other than class 3 (e.g. SIM specific messages), ME may choose not to route messages of such data coding schemes into TE (indication of a stored CBM may be given as defined in **<bm>**=1). 3 Class 3 CBMs are routed directly to TE using unsolicited result codes defined in <bm>=2. If CBM storage is supported, messages of other classes result in indication as defined in **<bm>**=1. **<ds>** 0 No SMS-STATUS-REPORTs are routed to the TE. 1 SMS-STATUS-REPORTs routed to TE not supported. 2 indication of memory location routed to TE not supported. **<bfr>** 0 TA buffer of unsolicited result codes defined within this command is flushed to the TE when <mode> 1...3 is entered (OK response shall be given before flushing the codes). 1 TA buffer of unsolicited result codes defined within this command is cleared when <mode> 1...3 is entered.
	Unsolicited result code **+CMTI: <mem>,<index>** Indication that new message has been received **+CBMI: <mem>,<index>** Indication that new CB-message has been received **+CMT: ,<length><CR><LF><pdu>** Short message is output directly **+CBM: <length><CR><LF><pdu>** Cell broadcast message is output directly
Reference GSM 07.05	Note Parameters can only be set to provider supported values

5.1.11 AT+CPMS Preferred SMS message storage

Test command AT+CPMS=?	Response +CPMS: (list of supported <mem1>s),(list of supported <mem2>s) ,(list of supported <mem3>s) Parameter see set command
Read command AT+CPMS?	Response +CPMS: <mem1>,<used1>,<total1>,<mem2>,<used2>,<total2>, <mem3>,<used3>,<total3> OK If error is related to ME functionality: +CMS ERROR Parameter see set command
Set command AT+CPMS = <mem1> [,<mem2> [,<mem3>]]	Response TA selects memory storages <mem1>, <mem2> and <mem3> to be used for reading, writing, etc. +CPMS: <used1>,<total1>,<used2>,<total2>,<used3>,<total3> OK If error is related to ME functionality: +CMS ERROR:<err> Parameter <mem1> Messages to be read and deleted from this memory storage "SM" SIM message storage <mem2> Messages will be written and sent to this memory storage "SM" SIM message storage <mem3> Received messages will be placed in this memory storage if routing to PC is not set ("+CNMI") "SM" SIM message storage <usedx> Number of messages currently in <memx> <totalx> Number of messages storable in <memx>
Reference GSM 07.05	Note

5.1.12 AT+CSCA SMS service centre address

Test command AT+CSCA=?	Response **OK**
Read command AT+CSCA?	Response **+CSCA: <sca>,<tosca> OK** Parameter see set command
Set command AT+CSCA = <sca>[,<tosca>]	Response TA updates the SMSC address, through which mobile originated SMs are transmitted. In text mode, setting is used by send and write commands. In PDU mode, setting is used by the same commands, but only when the length of the SMSC address coded into **<pdu>** parameter equals zero. *Note:* this command writes the service centre address to non-volatile memory. **OK** Parameter **<sca>** GSM 04.11 RP SC address Address-Value field in string format; BCD numbers (or GSM default alphabet characters) are converted into characters; type of address given by **<tosca>** **<tosca>** Service centre address format GSM 04.11 RP SC address Type-of-Address octet in integer format (default refer **<toda>**) *Note:* Parameter field <tosca> is ignored, national/international call center numbers are recognized by the leading + in the number.
Reference GSM 07.05	Note

5.1.13 AT+CSCB Select cell broadcast messages

Test command AT+CSCB=?	Response `+CSCB: (list of supported <mode>s)` Parameter `<mode>` 0 Accepts messages that are defined in <mids> and <dcss> 1 Does not accept messages that are defined in <mids> and <dcss>
Read command AT+CSCB? Write command AT+CSCB=[<mode>[,<mids>[,<dcss>]]]	Response `+CSCB: <mode>,<mids>,<dcss>` Parameter `<mode>` See Test command `<mids>` String type; combinations of CBM message IDs `<dcss>` String type; combinations of CBM data coding schemes
Reference GSM 07.05	Note It is possible that this command will be changed in case of deviation from U35!

5.1.14 AT+CSDH Show SMS text mode parameters

Test command AT+CSDH=?	Response **+CSDH:** (list of supported **<show>**s) **OK** Parameter see set command
Read command AT+CSDH?	Response **+CSDH:<show> OK** Parameter see set command
Set command AT+CSDH= <show>	Response TA sets whether or not detailed header information is shown in text mode result codes. **OK** Parameter **<show>** **0** do not show header values defined in commands **+CSCA** and **+CSMP** (**<sca>**, **<tosca>**, **<fo>**, **<vp>**, **<pid>** and **<dcs>**) nor **<length>**, **<toda>** or **<tooa>** in **+CMT**, **+CMGL**, **+CMGR** result codes for SMS-DELIVERs and SMS-SUBMITs in text mode; for SMS-COMMANDs in **+CMGR** result code, do not show **<pid>**, **<mn>**, **<da>**, **<toda>**, **<length>** or **<cdata>** **1** show the values in result codes
Reference GSM 07.05	Note

5.1.15 AT+CSMP Set SMS text mode parameters

Test command AT+CSMP=?	Response **+CSMP:** (list of supported **<fo>**s), (list of supported **<vp>**s) **OK** Parameter see set command
Read command AT+CSMP?	Response **+CSMP:<fo>,<vp> OK** Parameter see set command
Set command AT+CSMP= [<fo>[<vp>[,pid> [,<dcs>]]]	Response TA selects values for additional parameters needed when SM is sent to the network or placed in a storage when text format message mode is selected. It is possible to set the validity period starting from when the SM is received by the SMSC (**<vp>** is in range 0... 255) or define the absolute time of the validity period termination (**<vp>** is a string). Parameter **<fo>** depending on the command or result code: first octet of GSM 03.40 SMS-DELIVER, SMS-SUBMIT (default 17), , or SMS-COMMAND (default 2) in integer format **<vp>** depending on SMS-SUBMIT <fo> setting: GSM 03.40 TP-Validity-Period either in integer format (default 167) **<pid>** Protocol-Identifier in integer format (default 0), refer GSM 03.40 **<dcs>** SMS Data Coding Scheme (default 0), or Cell Broadcast Data Coding Scheme in integer format depending on the command or result code: GSM 03.38
Reference GSM 07.05	Note The command writes the parameters in NON-VOLATILE memory.

5.1.16 AT+CSMS Select Message Service	
Test command AT+CSMS=?	Response +CSMS: (list of supported <service>s) OK Parameter see set command
Read command AT+CSMS?	Response +CSMS: <service>,<mt>,<mo>,<bm> OK Parameter see set command
Set command AT+CSMS= <service>	Response +CSMS: <mt>,<mo>,<bm> OK If error is related to ME functionality: +CMS ERROR: <err> Parameter <service> **0** GSM 03.40 and 03.41 (the syntax of SMS AT commands is compatible with GSM 07.05 Phase 2 version 4.7.0; Phase 2+ features which do not require new command syntax may be supported (e.g. correct routing of messages with new Phase 2+ data coding schemes)) **1** GSM 03.40 and 03.41 (the syntax of SMS AT commands is compatible with GSM 07.05 Phase 2+ version; the requirement of <service> setting 1 is mentioned under corresponding command descriptions) Currently not available with the TC35. 128 Compatibility to Phase 1 and to device type M1 (manufacturer specific) <mt> Mobile Terminated Messages: **0** Type not supported **1** Type supported <mo> Mobile Originated Messages: **0** Type not supported **1** Type supported <bm> Broadcast Type Messages: **0** Type not supported **1** Type supported
Reference GSM 07.05	Note