T.C.
HALİÇ UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES

THESIS FOR THE DEGREE OF MASTER OF SCIENCE IN
COMPUTER ENGINEERING PROGRAM

MICROCONTROLLER BASED TRACKING SYSTEM

PREPARED BY
Fatih ERTÜRK

SUPERVISOR
Yrd.Doç.Dr. Yüksel BAL
January, 2008
ISTANBUL

TABLE OF CONTENT

TABLE OF CONTENT I
TABLE OF FIGURE II
SUMMARY IV
ÖZET V
ACKNOWLEDGMENTS VI
INTRODUCTION 1
MICROCONTROLLER BASED TRACKING DEVICES UNITS 1
1. GPS (GLOBAL POSITON SYSTEMS) 2
1.1 What is GPS (Global Position System) 2
1.2 The Three Segment of GPS 2
1.2.1 Space Segmet 3
1.2.2 The Control Segment 4
1.2.3 The User Segment 6
1.3 How Does The Global Positioning Sytem Works 6
1.3.1 Time is of Essence 7
1.3.2 Coming Full Circle 8
1.3.3 Almanac Data 8
1.4 GPS System Operation 8
1.4.1 Determining Your Position 9
1.4.2 Uses of GPS Technology 9
1.4.3 Location 10
1.4.4 Navigation 10
1.4.5 Tracking 10
1.4.6 Mapping 10
1.4.7 Timing 11
1.4.8 Measuring Your Distance 11
1.4.9 Error Correction 12
1.4.10 Differential GPS 13
1.4.11 Carrier-Phase GPS 15
1.4.12 GPS Competitors 16
1.4.13 Glonass 16
1.4.14 Galileo 17
1.4.15 GPS Smart Receiver Mouse 17
1.5 Common NMEA Sentence Types 18
1.5.1 GPS $GPRMC Sentences 19

2. MICROCONTROLLER UNIT 20
2.1 Description of a Microcontroller 20
2.2 The Task of The Microcontroller Unit in Tracking System 20
2.3 How to Choose The Right Microcontroller 20
2.4 Microcontroller Specification; 21
2.5 Description of Microcontroller 22
2.6 Pin Description 24
2.6.1 Crystal Oscillator 25
2.7 Architectural Overview 26
2.8 General Purpose Register Files 29
2.8.1 X-register, Y-register, and Z-register 30

2.8.2 ALU - Arithmetic Logic Unit 30
2.8.3 In-System Programmable Flash Program Memory 30
2.8.4 EEPROM Data Memory 31
2.4 Serial Communication 31
2.4.1 Serial Communication General Concept 31

3. GSM (GLOBAL SYSTEM FOR MOBILE COMMUNICATIONS) 34
3.1 GSM Modem Unit 35
3.1.1. General GSM Systems Information 35
3.1.2 SMS (Short Messaging Service) for GMS Systems 36
3.2 Used GSM (Global System for Mobile Communication) Modem of Thesis 37
3.2.1 Wavecom Fastrack Modem M13 Series 37
3.2.1.1 General Specification 38
3.2.2 Basic Features and Services 39
3.2.3 Connectivity Control and Main AT Command of GSM Modem 39
3.2.4 Functional Description 43
3.2.4.1 Architecture 43
3.2.5 RS 232 Serial Link 44
3.3 Wismo Quick 2406B Series GSM Modul 45
3.3.1 General Specification 45
3.3.2 Functional Description 47
3.3.3 RF Baseband Functionalities 48

4. OPEN AT SOFTWARE DEVELOPMENT 49
4.1 Software Architecture 49
4.1.1 Minimum Embedded Application Code 50
4.1.2 Specificity of AT Commonds in The Open AT Architecture 50
4.1.2.1 AT Command Size 50
4.1.2.2 AT+WDWL Command 50
4.1.2.3 AT+WOPEN Command 51
4.2 Memory Management 52
4.3 Command Pre-Parsing Limitation 53
4.4 Missing Unsolicited Messages in Remote Application 53
4.4.1 Minimum Embedded Application Code 53
4.5 Security 54
4.5.1 Software Security 54
4.5.2 RAM Access Protection 54
4.5.3 Watchdog Protection 55
4.5.4 Hardware Security 55

5. THESIS APPLICATION 56
5.1 How to Set GPS Mouse Receiver 57
5.2 Wavecom GSM Modem Command 58
5.3 SOFTWARE ON ATMEL 90S2313 MICROCONTROLLER 59
5.4 PROGRAMMABLE GSM MODULE 66

REFERENCES 76

TABLE OF FIGURE

Figure 1	The Three Segment of GPS	3
Figure 2	Space Segment	4
Figure 3	The Control Segment	5
Figure 4	Defense Satallite Communication System	6
Figure 5	The Two Possible Location	7
Figure 6	GPS Smart Receiver Mouse	17
Figure 7	The AT 90S2313 Block Diagram	23
Figure 8	AT90S2313 Pin Description	24
Figure 9	Internal Osilator Connection	25
Figure 10	External Osilator Connection	26
Figure 11	The AT90S2313 AVR RISC Architecture	27
Figure 12	Memory Map	28
Figure 13	AVR CPU General Purpose Working Registers	29
Figure 14	X, Y, and Z Registers	30
Figure 15	Functional Architecture of Wavecom Fastrack	43
Figure 16	RS 232 Connections	44
Figure 17	Functional Description Vismo GSM Module	47
Figure 18	Software Architecture	49
Figure 19	Resetting GSM Module	55
Figure 20	Designed Atmel Microcontroller Unit Block Diagram	56
Figure 21	NemeriX GPS Control Tool	57
Figure 22	Thesis Device Picture	74

T.C.
HALİÇ UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES IN
PURE AND APPLIED SCIENCES

THESIS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER ENGINEERING PROGRAM

Microcontroller Based Tracking System

Prepared By

Fatih ERTÜRK

SUPERVISOR

Yrd Doç Dr. Yüksel BAL

January, 2008

SUMMARY

In this thesis a microcontroller based tracking system is proposed and implemented. The designed system checks the position and velocity of the vehicle by using GPS and then send the valid GPS data to a GSM telephone by using GSM network. Nowadays, vehicle tracking system can be applied easly.The speed, position and others. information of the vehicle can be obtained in real time. The data obtained can be shown on screen text form or in graphic. Every land, sea and air vehicle's position all over tha world can be followed on digital map by using satallities so that application area of this system is a large range. Morover transport filos, rent a car companies can use this system easly,

Keywords: GPS, Tracking, GSM, Microcontroller,

T.C.
HALİÇ ÜNİVERSİTESİ
FENBİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Mikrodenetleyici Tabanlı İzleme sistemi

Hazırlayan

Fatih ERTÜRK

Danışman

Yrd.Doç.Dr. Yüksel BAL

Ocak, 2008

ÖZET

Bu tez çalışmasında bir araç takip sistemi önerilmiş ve gerçekleştirilmiştir. Sistem aracın hız ve yer bilgilerini araçta bulunan GPS alıcısı yoluyla elde eder ve bu bilgileri bir mikrodenetleyici devresi ve uygun bir kod ile alıcı GSM telefonuna aktarır. Bu telefona bağlı bilgisayarda bulunan sayısal harita üzerinde araç konumu işaretlenir. Sistem oldukça güvenilir şekilde çalışmaktadır. Dünya üzerinde kara, hava ve deniz araçları bu sistem ile sayısal haritalar üzerinde uydular aracılığı ile izlenebilir hale gelmekle birlikte taşımacılık, araç kiralama vb. sektörlerde verimliliği artırmak amacı ile kullanılmaktadır.

ACKNOWLEDGMENTS

I would like to start by thanking Yrd.Doç Dr. Yüksel BAL for his encouragement and guidance through out this project.

INTRODUCTION

MICROCONTROLLER BASED TRACKING DEVICES UNITS

In the silicon industry there are so many GPS device, microcontrollers and GSM modules producing companies that choosing the right products from their hundreds in their catalog is as hard as to work on them. Therefore, while choosing GPS receiver, microcontroller unit and GSM modem unit the key factor is that you have to know your needs to make it possible for you to get the right one.

At this Project ,Mobile unit consist of GPS Mouse receiver which is receive data from GPS satellites and send NMEA protocol based signal by RS 232, Microcontroller (Atmel 90s2313) receive the GPS data and filtering valid data,and send data by using AT+GPS command ,then wavecom GSM&GPRS modem send this valid data, dedicated time range or if request last location send GPS data by using SMS to central unit. Components specification detail explained at follow;

1. GPS (GLOBAL POSITON SYSTEMS)

1.1 What is GPS (Global Position System)

Global Positioning System, a network of satellites that continuously transmit coded information, which makes it possible to precisely identify locations on earth by measuring distance from the satellites.

As a stated in the defination above, GPS stands for Global Positioning System, and refers to group of U.S: Department of Defense satellites constantly circling the earth .The satellites transmit very low power radio signals allowing anyone with a GPS receivers to determine their location on Earth.The remarkable system was not cheap to build ,costing the U.S. billions of dollars. Ongoing maintenance, including the launch of replacement satellites,adds to costs of the system. Amazingly, GPS actually predates the introduciton of personel computer. ITS designers may not have foreseen a data we would be carrying small portable receivers,weighing less than a pound, that would not only tell us where we are in position coordinates(latitude/longitude), but would even display our lacation on an electronic map along with cities,streets and more.

1.2 The Three Segment of GPS

The NAVSTAR system (the acronym for Navigation Satallite Timing and Ranging ,the official U.S. Department of Defense name for GPS) as shown Figure 1 consists of a space segment (the satellites), a control segment (the ground stations), and user segmetn(GPS receiver).

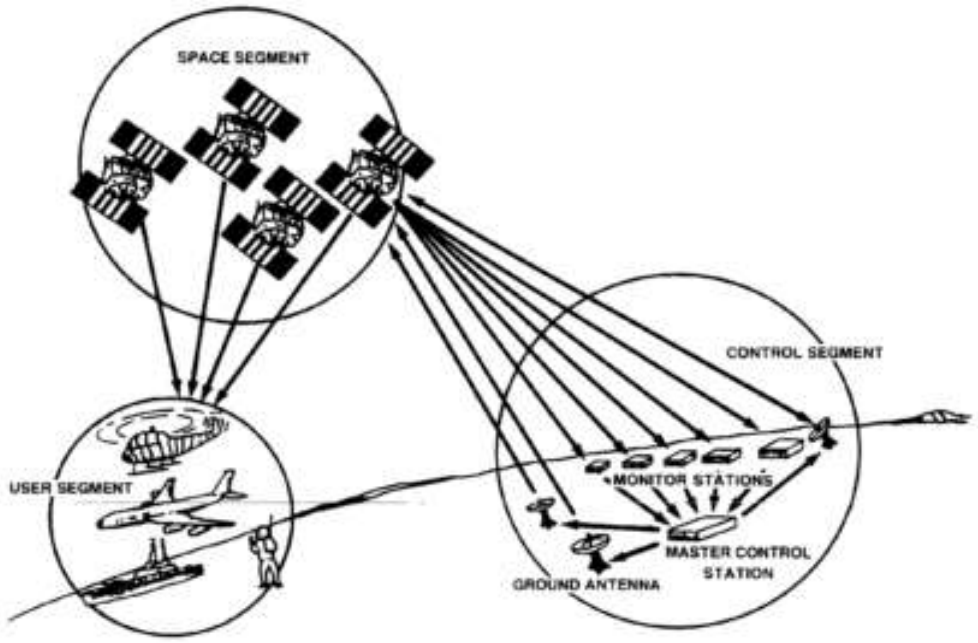


Figure 1-The Treee Segment of GPS

1.2.1 Space Segmet

The Space segment ,which consists of at least 24 satellites (21 Active plus 3 operating spares) is the hearth of the system.The satellites are in what's called a " high orbit" aboute 12.000 miles above the Earth's surface.Operating at such a high altitude allow a the signals to cover a greater area.The satellites are arranged in their orbits so a GPS receiver on earth can always receive from at least four of them at any given time.

The satellites are travelling at speeds of 7.000 miles an hour ,which allows them to circle the earth once every 12 hours. They are powered by solar energy and a rebuilt about 10 years. If the solar energy fails, they have a backup batteries to keep flying in the correct path.

Each satellites transmit low power radio signals on several frequencies (designated L1,L2, etc) Civilian GPS receivers"listen" on the L1 frequency of 1575,42 Mhz in the UHF band.The signal travels "line of sight",meaning it will pass through clouds,glass and plastic, but will not go through most solid objects such as buildings and mountains.

L1 contains two "pseudorandom"(a complex pattern of digital code)signals, the protected(P) code and the Coarse/Acquisition(C/A) code.Each satellite transmits a unique code, allowing the GPS receiver to identify the signals."Anti-

spoofing"refers to the scrambling of the P code in order to prevent its unauthorized Access.The P code is also called the"P(Y)" or "Y" code.

The main purpose of these coded signals is to allow for calculating the travel time from the satellite to GPS receiver on the Earth .This travel time is also called the Time of Arrival.The Travel time multiplied by the speed of light equals the satellite range(distance from the satellites to the GPS receiver) The Navigation message (the information the satellites transmit to a receiver) contains the satellite orbital and clock information and general system status messages and an ionospheric delay model.The satellite signals are timed using highly accurate atomic clocks.

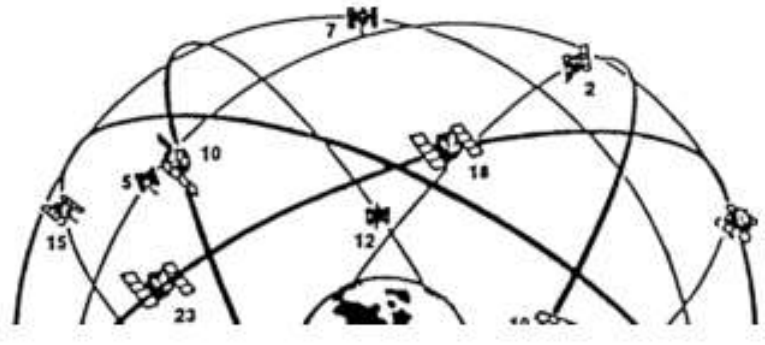


Figure 2.Space Segment

1.2.2 The Control Segment

The "control" segment does what its name implies GPS satellites by tracking them and then providing them with corrected orbital and clock (time)information. There are five control stations located around the world ,four unmaneged receiving stations constantly receive data from the satellites and then send that information to the master control station.The master control station "corrects" the satellite data and, together with two other antenna sites, send "uplinks"the information to the GPS satellites.

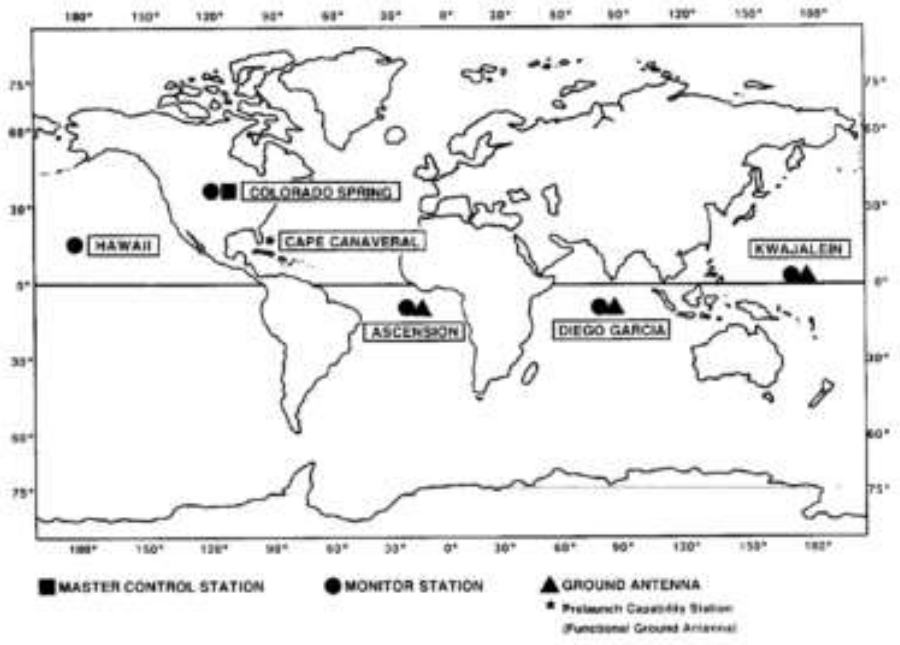


Figure 3 The Control Segment

The MCS is the central processing facility for the Control Segment and is responsible for monitoring and managing the satellite constellation. The MCS functions include control of satellite station keeping maneuvers, reconfiguration of redundant satellite equipment, regularlyupdating the navigation messages transmitted by the satellites, and various other satellite health monitoring and maintenance activities. The monitor stations passively track all GPS satellites in view, collecting ranging data from each satellite. This information is transmitted to the MCS where the satellite ephemeris and clock parameters are estimated and predicted. The MCS uses the ground antennas to periodically upload the ephemeris and clock data to each satellite for retransmission in the navigation message. Communications between the MCS the MS and GA are typically accomplished via the U.S. Defense Satellite Communication System (DSCS). The navigation message update function is graphically depicted in Figure 4

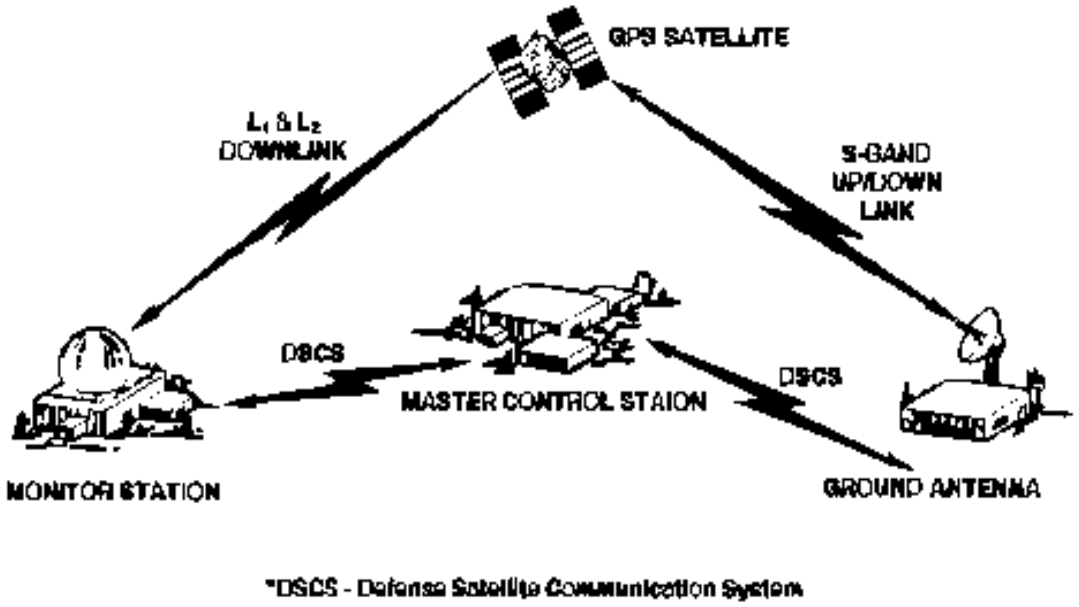


Figure 4 Defense Satallite Communication System

1.2.3 The User Segment

The User segmet simply consists of you and yur GPS receiver. As mentioned previously ,the use segment consists of boaters,pilots,hikers,hunters, the military and anyone else who wants to know where they are, where they have been or where they are going.

1.3 How Does The Global Positioning Sytem Works

The GPS receiver has to know two things if it's going to do its job. It has to know WHERE the satellites are (location) and how FAR AWAY they are(distance).

Let's first look at how the GPS receiver knows where the satellites are located in space. The GPS receiver picks up two kinds of coded information from the satellites. One type of infomation ,called"almanac" data,contaions the approximate position (location) of the satellite. This data is continuously trasnmitted and stored in the memory of the GPS receiver so it knows the orbits of satellites and where each satellite is supposed to be. The almanac data is periodically updated with new informations the satellites move around. Any Satellites can travel slightly out of orbit ,so ground monitor stations keep track of the satellite orbits,altitude,location, and speed.The ground stations corrected data up to the satellites. This corrected and exact position data is called the "ephemeris" data ,which is valid for about four to six hour

,and is transmitted in the coded information to theGPS receiver.So, having received the almanac and ephemeris data,the GPS receiver knows the position (location) of the satellites at all times.

1.3.1 Time is of Essence

Eventhough the GPS receiver knows the precise location of the satellites in space, it stil needs know how far away the satellites are (the distance) so it can determine its position on Earth.There is a simple ormula that tells the receiver how far it is from each satellite, Your Distance from a given satellite objectequals the velocity of the transmitted signal multiplied by the time it takes the signal to reach you(Velocity x Travel Time=Distance).Remember as kid how you could find out how far a thunderstoem was from you ?When you saw a lightning flash you counted the number of seconds until you heard the thunder. The longer the count ,the further waya the storm was.GPS Works on the same principle, called"Time of Arrival"

Using the same basic Formula to dermine deistance ,the receiver already knows the velocity.It's the speed of a radio wave-186,00 miles per second (the speed of light), Less any delay as the signal travels through the Earth's atmosphere.

Now the GPS receiver needs to determine the time part of the Formula. The answer lies in the coded signals the satellites trasmit. The trasnmitted code is called "pseudo-random code" because it looks like a noise signal.When a satellite is generating the pseudo-random code, the GPS receiver is generating the same code and tries to match it up to the satellite's code.The receiver is generating the same code and tries to match it up to the satellite's code.Thr receiver then compares the two codes to determine how much it needs to delay (or Shift) its code to match the satellite code. This delay time (shift) is multiplied by the speed of light to ger the distance.

Your GPS receiver clock does not keep the time as precisely as the satellite clocks. Putting an atomic clock in your GPS receiver would make it much larger and far too expensive !So each distance measurement needs to be corrected to account for the GPS receiver's internal clock error.For this reason ,the range measurement is referrd to as a"pseudo-range1.To determine positon using pseudo-range data, minimum of four satellites must be tracked and the four fixes must be recomputed until the clock error disappears.

1.3.2 Coming Full Circle

Now that we have both satellite location and distance, receiver can determine a position. Let's say we are 11.000 miles from one satellite.Our location would be somewhere on an imaginary sphere that has the satellite in the center with radius of 11,000 miles.Then let's say we are 12,000 miles from another satellite.The seconf sphere woul intersect the first sphere to create a common circle.If we add a third satellite,at a distance of 13.000 miles, we now have two common points where the three spheres intersect.

Even Though there are two possible positions,they differ greatly in latitude/longitude position AND altitude.To determne which of the two common points is your actual position ,you'll need to enter your approxmiate altitude into the GPS receiver. This will allow the receiver to calculate a two –demensional position (latitude,longitude). However,by adding a fourth satellite, the receiver can determine your three-dimensional position(latitude, longitude,altidude). Let's say our distance from a fourth satellite is 10,ooo mmiles . we now have a fourth sphere intersecting the fisrt three spheres at one common point.

1.3.3 Almanac Data

The unit stores data about where the satellites are located at any given time. This data is called the almanac. Sometimes when the GPS unit is not turned on for a length of time, the almanac can get outdated or"cold".

When the GPS receiver is "cold" , it could take longer to acquire satellites. A receiver is considered "warm" when the data has been collected from the satellites within the last four to six hours.When you are looking for a GPS unit to buy ,yo uman see "cold" and "warm"acquisition time specifications. If the time it takes the GPS unit to lock on to the signals and calculate a position is important to you ,be sure to check the acquisition times.

1.4 GPS System Operation

The basic idea behind GPS is to use satellites in space as reference points for locations on earth. With GPS, signals from the satellites arrive at the exact position of the user and are triangulated. This triangulation is the key behind accurate location determining and is achieved through several steps.

1.4.1 Determining Your Position

Suppose we measure our distance from a satellite and find it to be 11,000 miles (how it is measured is covered later). Knowing that we're 11,000 miles from a particular satellite narrows down all the possible locations we could be in the whole universe to the surface of a sphere that is centred on this satellite and has a radius of 11,000 miles.

Next, say we measure our distance to a second satellite and find out that it's 12,000 miles away. That tells us that we're not only on the first sphere but we're also on a sphere that's 12,000 miles from the second satellite, i.e. somewhere on the circle where these two spheres intersect. If we then make a measurement from a third satellite and find that we're 13,000 miles from that one, that narrows our position down even further, to the two points where the 13,000 mile sphere cuts through the circle that's the intersection of the first two spheres.

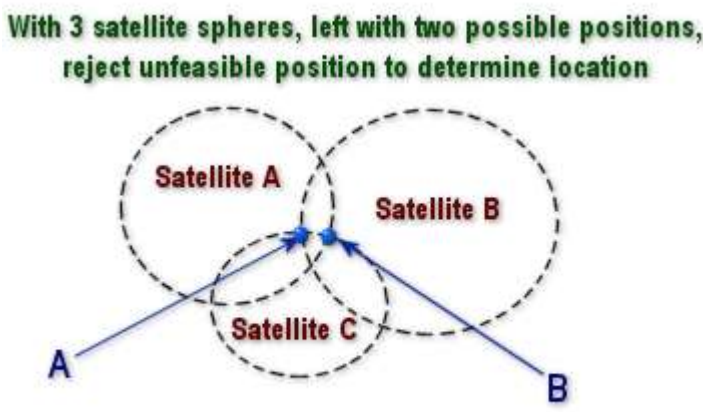


Figure 5 The two possible locations

1.4.2 Uses of GPS Technology

GPS technology has matured into a resource that goes far beyond its original design goals. These days people from a plethora of professions are using GPS in ways that make their work more productive, safer, and sometimes even easier. There are five main uses of GPS today:

1. Location- *determining a basic position*.
2. Navigation - *getting from one location to another*.
3. Tracking - *monitoring the movement of people and things.*

4. Mapping- *creating maps.*
5. Timing - *providing precise timing* .

1.4.3 Location

The first and most obvious application of any Location Based Service such as GPS is the simple determination of a "position" or location. GPS was the first positioning system to offer highly precise location data for any point on the planet, in any weather. Knowing the precise location of something, or someone, is especially critical when the consequences of inaccurate data are measured in human terms.

1.4.4 Navigation

GPS helps you determine exactly where you are, but sometimes it is more necessary to know how to get somewhere else. Recall that GPS was originally designed to provide navigation information for ships and planes. So it's no surprise that while this technology is appropriate for navigating on water, it's also very useful in the air and on the land.

1.4.5 Tracking

GPS used in conjunction with communication links and computers can provide the backbone for systems tailored to applications in agriculture, mass transit, urban delivery, public safety, and vessel and vehicle tracking. Therefore , more and more police, ambulance, and fire departments are adopting systems like GPS-based AVL (Automatic Vehicle Location) Manager to pinpoint both the location of the emergency and the location of the nearest response vehicle on a computer map. With this kind of clear visual picture of the situation, dispatchers can react immediately and effectively.

1.4.6 Mapping

Mapping the planet has never been an easy task, but GPS today is being used to survey and map it precisely, saving time and money in this most stringent of all applications. GPS can help generate maps and models of everything in the world , mountains, sea, rivers, cities and help manage endangered animals,archaeological treasures, precious minerals and all sorts of resources , as well as accurately managing the effect of damage and disasters.

For example peace brokers used GPS generated maps to determine the partitions of Bosnia under the Dayton Peace Accord, and GPS was used to map Cambodia accurately for the UN stabilisation force introduced after the civil war ended in 1993. In fact large tracks of Cambodia had never been mapped at all, due to inhospitable terrain and low population density. GPS solved this in weeks, whereas it would have taken months prior to the introduction of GPS.

1.4.7 Timing

GPS can also be used to determine precise time, time intervals, and frequency. There are three fundamental ways we use time:

- As a universal marker,
- As a way to synchronize
- To provide an accurate, unambiguous sense of duration.

As discussed before, GPS satellites carry highly accurate atomic clocks. And in order for the system to work, GPS receivers here on the ground synchronize themselves to these clocks. That means that every GPS receiver is, in essence, an atomic accuracy clock. Astronomers, power companies, computer networks, communications systems, banks, and radio and television stations can benefit from this precise timing.

So by ranging from three satellites we can narrow our position to just two points in space. To decide which one is our true location we could make a fourth measurement. But usually one of the two points is a ridiculous answer (either too far from Earth or moving at an impossible velocity) and therefore can be rejected without a measurement.

1.4.8 Measuring Your Distance

How the satellites actually measure the distance is quite different from determining your position and essentially involves using the travel time of a radio message from the satellite to a ground receiver. To make the measurement we assume that both the satellite and our receiver are generating the same pseudo-random code at exactly the same time. This pseudo-random code is a digital code unique to each satellite , designed to be complex enough to ensure that the receiver doesn't accidentally sync up to some other signal. Since each satellite has its own

unique Pseudo-Random Code this complexity also guarantees that the receiver won't accidentally pick up another satellite's signal. So all the satellites can use the same frequency without jamming each other. And it makes it more difficult for a hostile force to jam the system, as well as giving the DOD a way to control access to the system.

By comparing how late the satellite's pseudo-random code appears compared to our receiver's code, we determine how long it took to reach us. Multiply that travel time by the speed of light and you obtain the distance between the receiver and the satellite. However this calls for precise timing to determine the interval between the code being generated at the receiver and received from space. On the satellite side, timing is almost perfect due to their atomic clocks installed within each satellite. However as it would be extremely uneconomical for receiver to use atomic clocks a different method must be found.

GPS solves this problem by using an extra satellite measurement for the following reason: If our receiver's clocks were perfect, then all our satellite ranges would intersect at a single point - our position. But with imperfect clocks, a fourth measurement, will not intersect with the first three satellite ranges. So the receiver's computer will then calculate a single correction factor that it can subtract from all its timing measurements that would cause them all to intersect at a single point. That correction brings the receiver's clock back into sync with universal time , ensuring (once the correction is applied to all the rest of the receivers' measurements) precise positioning.

1.4.9 Error Correction

As would be expected, a variety of different errors can occur within the system, some of which are natural, whilst others are artificial. First of all, a basic assumption, the speed of light, is not constant as this value changes as the satellite signals travel through the atmosphere. As a GPS signal passes through the charged particles of the ionosphere and then through the water vapour of the troposphere it gets slowed down, and this creates the same kind of error as bad clocks. This problem is tackled by attempting to use modelling of the atmospheric conditions of the day, and using dual-frequency measurement, i.e. comparing the relative speeds of two different signals. Another problem is multipath error , this is when the signal may bounce off various local obstructions before it gets to our receiver. Sophisticated

signal rejection techniques are used to minimize this problem.There are also potential problems at the satellites. Minute time differences can occur within the on-board atomic clocks, and sometimes position (*ephemeris)* errors can occur. These other errors can be magnified by a high GDOP "*Geometric Dilution of Precision*" This is where a receiver picks satellites that are close together in the sky, meaning the intersecting circles that define a position will cross at very shallow angles. That increases the grey area or error margin around a position. If the receiver picks satellites that are widely separated the circles intersect at almost right angles and that minimises the error region. Obviously good receivers determine which satellites will give the lowest GDOP.

Finally up to recently there was another , man-made source of errors. The U.S. was very mindful of the fact that terrorists and unfriendly governments could use the accurate positioning provided by GPS and so intentionally degraded GPS's accuracy. This policy is called *Selective Availability* or SA. This involves the DOD introducing some "noise" into the satellite's clock data which, in turn, adds noise (or inaccuracy) into position calculations. The DOD may also has been sending slightly erroneous orbital data to the satellites which they transmit back to receivers on the ground as part of a status message. Together these factors made SA the biggest single source of inaccuracy in the system. Military receivers used a decryption key to remove the SA errors and so they were considerably more accurate.However, effective May 2, 2000 selective availability has been eliminated. The recent terrorist attacks on America have not changed this position. This is due to the fact that civilian uses of GPS have become critical across the world, and because the United States Department of Defence now has the technology to localise the control system to deny GPS signals to selected areas.

1.4.10 Differential GPS

Using a modified form of GPS called *Differential GPS* (originally initiated by the U.S. Coast Guard to counter the accuracy degradation caused by Selective Availability) can significantly reduce the above errors. Even with SA eliminated, DGPS continues to be a key tool for highly precise navigation on land and sea. DGPS can yield measurements accurate to a couple of meters in moving applications and even better in stationary situations. Differential GPS involves the co-operation of

two receivers, one that's stationary and another that's roving around making position measurements.

As each GPS receivers use timing signals from at least four satellites to establish a position then each of those timing signals is going to have some error or delay depending on what sort of problems have occurred it on its journey down to Earth. Since each of the timing signals that go into a position calculation has some error, that calculation is going to be a compounding of those errors.

However if two receivers are fairly close to each other, say within a few hundred kilometres, the signals that reach both of them will have travelled through virtually the same slice of atmosphere, and so will have virtually the same errors

This means that you could use have one receiver to measure the timing errors and then provide correction information to the other receivers that are roving around. This allows virtually all errors to be eliminated from the system.

The reference station operates by receiving the same GPS signals as the roving receiver but instead of working like a normal GPS receiver it uses its known position to calculate timing, rather than using timing signals to calculate position. Essentially determining what the travel time of the GPS signals should be, and compares it with what they actually are. The difference is an "error correction" factor. The receiver then transmits this error information to the roving receiver so it can use it to correct its measurements.

Since the reference receiver has no way of knowing which of the many available satellites a roving receiver might be using to calculate its position, the reference receiver quickly runs through all the visible satellites and computes each of their errors. Then it encodes this information into a standard format and transmits it to the roving receivers. The roving receivers can then apply the corrections for particular satellites they are using. The United States Coast Guard and other international agencies are establishing reference stations all over the place, especially around busy harbours and waterways.

There are also different kinds of DGPS, for use when users don’t need precise positioning immediately. This is termed Post Processing DGPS, and is used when the roving receiver just needs to record all of its measured positions and the exact time it

made each measurement. Then later, this data can be merged with corrections recorded at a reference receiver for a final clean-up of the data, meaning you don't need the radio link required in real-time systems. Another form of DGPS, called Inverted DGPS, which is used to save money when operating a large fleet of users. With an inverted DGPS system the users would be equipped with standard GPS receivers and a transmitter and would transmit their standard GPS positions back to the tracking station (the main office). Then at the tracking station the corrections would be applied to the received positions.

1.4.11 Carrier-Phase GPS

This is a new version of GPS that can eliminate errors even better than other forms. Recall that a GPS receiver determines the travel time of a signal from a satellite by comparing the pseudo random code it's generating, with an identical code in the signal from the satellite. The receiver slides its code later and later in time until it syncs up with the satellite's code. The amount it has to slide the code is equal to the signal's travel time. The problem is that the bits (or cycles) of the pseudo random code are so wide that when the signals sync up there is room for error. Survey receivers are better as they start with the pseudo random code and then move on to measurements based on the carrier frequency for that code. This carrier frequency is much higher so its pulses are much closer together and therefore more accurate. At the speed of light the 1.57 GHz GPS signal has a wavelength of roughly twenty centimetres, so the carrier signal can act as a much more accurate reference than the pseudo random code by itself. And if it can get to within one percent of perfect phase like you expect with code-phase receivers you can (theoretically) obtain 3 or 4 millimetre accuracy.

In essence this method is counting the exact number of carrier cycles between the satellite and the receiver. The problem is that the carrier frequency is hard to count because it's so uniform. Every cycle looks like every other. The pseudo random code on the other hand is intentionally complex to make it easier to know which cycle you're looking at. But Carrier-phase GPS tackles this problem by using code-phase techniques to get close. If the code measurement can be made accurate to say, a meter, then we only have a few wavelengths of carrier to consider as we try to determine which cycle really marks the edge of our timing pulse. Resolving this carrier phase ambiguity for just a few cycles is a much more tractable problem and as the computers inside the receivers increase in processing power and functionality it's

becoming possible to make this kind of measurement without all the steps that survey receivers go through.

1.4.12 GPS Competitors

As it stands at the moment there are two satellite systems being marketed as current and future satellite systems billed as being able to provide location-based services.

1.4.13 Glonass

GLONASS is the Russian Federation's satellite navigation system. In many ways it is very similar to GPS: there are 24 satellites in the full constellation; each satellite transmits various data on two L-band carriers; there is one navigation signal that has been authorised for civilian use, and further navigation signals that have been reserved solely for Russian military use; there is a ground segment that monitors and control the satellites; and users passively receive its signals and are able to navigate with accuracy's of few tens of metres or better.

However, it is too easy from a user's perspective to presume that GLONASS is the same as GPS. Rather, there are some key differences between the two systems. In many ways, GLONASS is a more elegant and economical system design than GPS. Unfortunately there are a number of quality control issues and much more serious funding problems. The GLONASS system was declared fully operational on 18 January 1996. Since then there have been few occasions when the full complement of 24 satellites was operating. In addition to satellite outages eight satellites have been withdrawn from service since this date. So by late 1997 only 15 satellites were available for navigation.

Thankfully though, recent announcements by Russian officials may mean that 6 years of neglect of the GLONASS constellation of satellites will stop. Solid government financial commitments have been obtained to return the system to fully operational status by 2006, Russian officials say. Also, this new generation of GLONASS satellites (Glonass-M) will have longer in-orbit lives that should ease pressure on the system, they said. They are designed to have a seven-year service life, compared to three years for the current GLONASS. Whether this financial commitment will materialise is another thing, but even in its current weakened state,

GLONASS still has potential as a stand-alone navigation system, and as an augmentation to GPS.

1.4.14 Galileo

The Galileo satellite radio navigation system is an initiative launched by the European Union and the European Space Agency (ESA).The project architects plan deployment in 2006-7, becoming operational in 2008 at a yearly cost of €220m. It is projected by 2008 a constellation of 30 satellites should be available.The technology behind Galileo is designed to be more accurate and more reliable than GPS or GLONASS. This will allow safety-critical systems - such as air traffic control, and ship and car navigation - to be run on the technology. The system should also guarantee coverage to previously inaccessible areas such as those that are either blocked by buildings or isolated areas at high latitudes.However it is estimated that the Galileo project could cost more than €3bn ($2.6bn). With some of the member countries facing recession, now might not be the time to fund the idea. Britain and the Netherlands have already questioned the timing, and Germany has raised issues over costs. In contrast, France and Italy whose aerospace industries both stand to benefit from the system are looking forward to a go-ahead. The decision to push ahead with the project remains up in the air for now

1.4.15 GPS Smart Receiver Mouse

Figure 6 GPS Smart Receiver Mouse

RS 232- based communication,ultra low power design (38 mA, typical**)** The GPS smart receiver features the 16 channels. Ultra low power GPS architecture.This Complate enable GPS receiver provides high position, velocity and time accuracy performances as well as high sensitivity and tracking capabilities. The GPS receiver is ideal for many portable application such as PDA, Tablet PC, Smart phone, Personal Tracking, Vichle tracking etc.

Performance of receiver;

Frequency	L1,1575,42 Mhz
C/A Code	1.023 MHZ chip rate
Channel	16 channels all in view
Tracking Sensitivity	-152 dBm

1.5 Common NMEA Sentence Types

The following information describes the most common NMEA-0183 sentences transmitted by GPS receivers. The NMEA standard provides quite a range of sentences, but many relate to non-GPS devices and some others are GPS related but rarely used. We normally recommend the use of NMEA mode for new GPS applications to give maximum compatibility with all GPS receivers. Most GPS receivers also have a binary mode but it is normally best to reserve the use of binary GPS protocols for applications that really require their use, such as those requiring position updates of greater than once per second.

Sentence	**Description**
$GPGGA	Global positioning system fixed data
$GPGLL	Geographic position - latitude / longitude
$GPGSA	GNSS DOP and active satellites
$GPGSV	GNSS satellites in view
$GPRMC	Recommended minimum specific GNSS data
$GPVTG	Course over ground and ground speed

1.5.1 GPS $GPRMC Sentences

Used $GPRMC Sentence (Position and time) explanation

Example (signal not acquired):

$GPRMC,235947.000,V,0000.0000,N,00000.0000,E,,,041299,,*1D

Example (signal acquired):

$GPRMC,092204.999,A,4250.5589,S,14718.5084,E,0.00,89.68,211200,,*25

Field	**Example**	**Comments**
Sentence ID	$GPRMC	
UTC Time	092204.999	hhmmss.sss
Status	A	A = Valid, V = Invalid
Latitude	4250.5589	ddmm.mmmm
N/S Indicator	S	N = North, S = South
Longitude	14718.5084	dddmm.mmmm
E/W Indicator	E	E = East, W = West
Speed over ground	0.00	Knots
Course over ground	0.00	Degrees
UTC Date	211200	DDMMYY
Magnetic variation		Degrees
Magnetic variation		E = East, W = West
Checksum	*25	
Terminator	CR/LF	

2. MICROCONTROLLER UNIT

2.1 Description of a Microcontroller

A Microcontroller is higly integrated chip that contains all the components comprising a computer. Typically this includes a CPU ,RAM ,some from of ROM I/O ports, and timers.Unlike a general-purpose computer, which also includes all of these components,a microcontroller is designed for a very specific task which is to control a particular system. As a result,The parts can be simlified and reduced,which also cuts down on production costs. Systems using microcontrolelrs are sometimes one part of large device sor system so that they can be called embedded systems.

2.2 The Task of The Microcontroller Unit in Tracking System

The main task of the Microcontroller Unit in system is to manage the data input and output to from GPS modul and GSM modem.by using AT90S2313.The task the Microcontroller unit to receive data from GPS after some calculation ,to send the data to the GSM unit to be sent to the central unit.

2.3 How to Choose The Right Microcontroller

There are many Microcontroller or PIC (Peripheral interface Controller) ,While choosing microcontroller the key factory low cost, simple design.

Microcontroler based tracking systems Project, microcontroller chosen as processing unit is AVR ,Atmel 90S2313 which is 8 bit with 2 K Bytes of In-system programmable flash..

Product features;

2.4 Microcontroller Specification;

Utilizes AVR RISC Architecture

Architecture optimized for C-compiler

AVR-High-performance and low-power RISC Architecture

Data and Non-Volatile Program Memory

2 Kbyte of In-system Programmable Flash

128 Bytes of In-system programmble Eeprom

Peripheral Fetures

One 8-bit Timer/Counter with Separate Prescaler

One 16-bit Timer/Counter with Separate Prescaler,

Compare, Capture Modes and 8-, 9-, or 10-bit PWM

On-chip Analog Comparator

Programmable Watchdog Timer with On-chip Oscillator

SPI Serial Interface for In-System Programming

FullDuplexUART

Specifications

Low-power, High-speed CMOS Process Technology

Fully Static Operation

Power Consumption at 4 MHz, 3V, 25°C

Active: 2.8 mA

Idle Mode: 0.8 mA

Power-down Mode: <1 μA

I/O and Packages

15 Programmable I/O Lines

20-pin PDIP and SOIC

Operating Voltages

2.7 - 6.0V (AT90S2313-4)

4.0 - 6.0V (AT90S2313-10)

Speed Grades

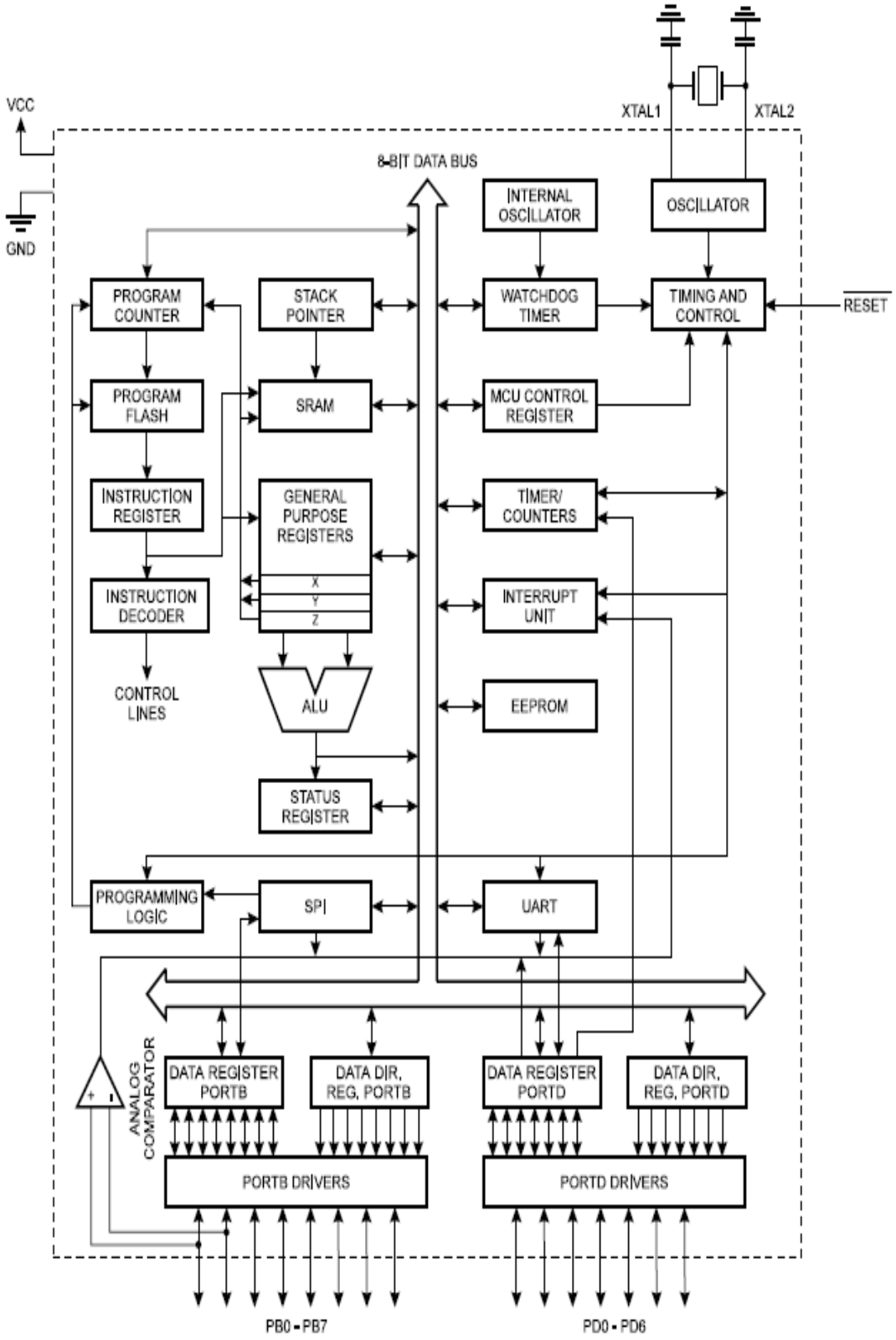
0 - 4 MHz (AT90S2313-4)

-0 - 10 MHz (AT90S2313-10)

2.5 Description of Microcontroller

The AT90S2313 is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. By executing powerful instructions in a single clock cycle, the AT90S2313 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed. The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.The AT90S2313 provides the following features: 2K bytes of In-System Programmable Flash, 128 bytes EEPROM, 128 bytes SRAM, 15 general purpose I/O lines, 32 general purpose working registers, flexible Timer/Counters with compare modes, internal and external interrupts, a programmable serial UART, programmable Watchdog Timer with internal Oscillator, an SPI serial port for Flash memory downloading and two software

Figure 7 The AT90S2313 Block Diagram



The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port and interrupt system to continue functioning . The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next external interrupt or Hardware Reset. The device is manufactured using Atmel's high-density non-volatile memory technology. The On-chip In-System Programmable Flash allows the Program memory to be reprogrammed in-system through an SPI serial interface or by a conventional non-volatile memory programmer. By combining an enhanced RISC 8-bit CPU with In-System Programmable Flash on a monolithic chip , the Atmel AT 9 0 S 2 3 1 3 is a powerful microcontroller that provides a highly flexible and cost-effective solution to many embedded control applications. The AT90S2313 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, In-Circuit Emulators and evaluation kits.

2.6 Pin Description

VCC Supply voltage pin.

GND Ground pin.

Port B (PB7..PB0)

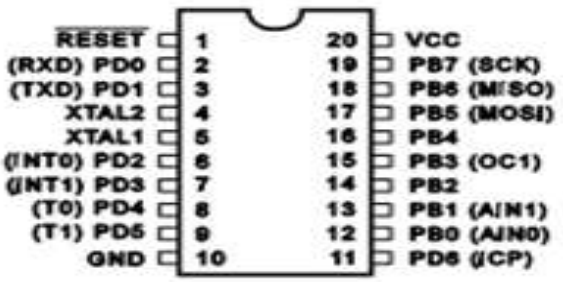


Figure 8 AT90S2313 Pin Description

Port B is an 8-bit bi-directional I/O port. Port pins can provide internal pull-up resistors (selected for each bit). PB0 and PB1 also serve as the positive input (AIN0) and the negative input (AIN1), respectively, of the On-chip Analog Comparator. The Port B output buffers can sink 20 mA and can drive LED displays directly. When pins PB0 to PB7 are used as inputs and are externally pulled low, they will source

current if the internal pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not active. Port B also serves the functions of various special features of the AT90S2313

Port D (PD6..PD0) Port D has seven bi-directional I/O ports with internal pull-up resistors, PD6..PD0. The Port D output buffers can sink 20 mA. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not active.

RESET Reset input. A low level on this pin for more than 50 ns will generate a Reset, even if theclock is not running. Shorter pulses are not guaranteed to generate a Reset.

XTAL1 Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

XTAL2 Output from the inverting Oscillator amplifier.

2.6.1 Crystal Oscillator

XTAL1 and XTAL2 are input and output, respectively, of an inverting amplifier that can be configured for use as an On-chip Oscillator, as shown in Figure 2. Either a quartz crystal or a ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 3

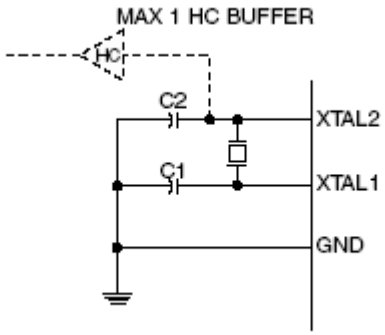


Figure 9 Internal Osilator Connections

When using the MCU Oscillator as a clock for an external device, an HC buffer should be connected as indicated in the figure.

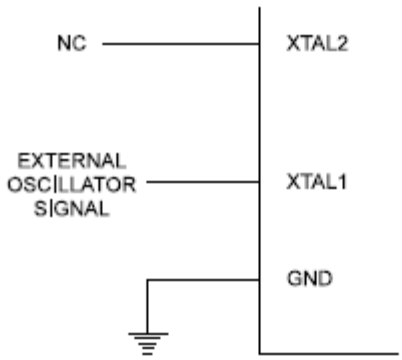


Figure 10 External Osilator Connection

2.7 Architectural Overview

The fast-access Register File concept contains 32 x 8-bit general purpose working registers with a single clock cycle access time. This means that during one single clock cycle, one ALU (Arithmetic Logic Unit) operation is executed. Two operands are output from the Register File, the operation is executed, and the result is stored back in the Register File - in one clock cycle.

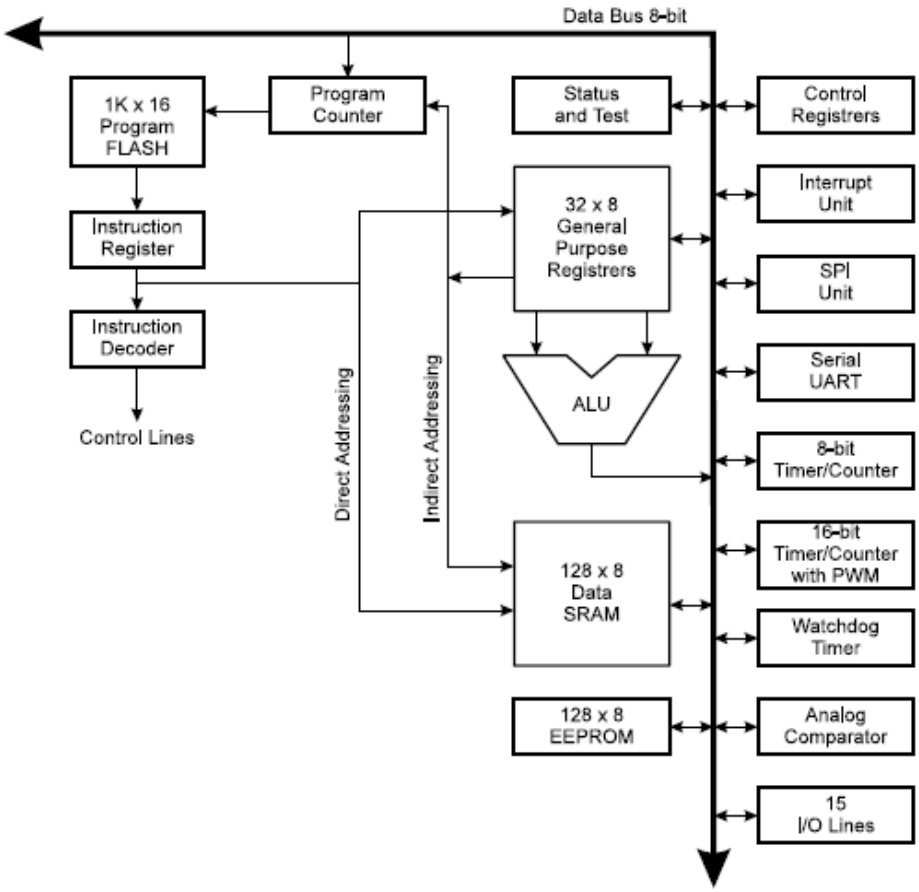


Figure 11. The AT90S2313 AVR RISC Architecture

Six of the 32 registers can be used as three 16-bit indirect address register pointers for Data Space addressing - enabling efficient address calculations. One of the three address pointers is also used as the address pointer for the constant table look-up function. These added function registers are the 16-bit X-register, Y-register, and Z-register.The ALU supports arithmetic and logic functions between registers or between a constant and a register. Single register operations are also executed in the ALU. Figure 4 shows the AT90S2313 AVR RISC microcontroller architecture.

In addition to the register operation, the conventional memory addressing modes can be used on the Register File as well. This is enabled by the fact that the Register File is assigned the 32 lowermost Data Space addresses ($00 - $1F), allowing them to be accessed as though they were ordinary memory locations.

The I/O memory space contains 64 addresses for CPU peripheral functions such as control registers, Timer/Counters, A/D converters and other I/O functions. The I/O memory can be accessed directly or as the Data Space locations following those of the Register File, $20 - $5F. The AVR has Harvard architecture - with

separate memories and buses for program and data. The program memory is accessed with a 2-stage pipeline. While one instruction is being executed, the next instruction is pre-fetched from the program memory. This concept enables instructions to be executed in every clock cycle. The program memory is In-System Programmable Flash memory. With the relative jump and call instructions, the whole 1K address space is directly accessed . Most AVR instructions have a single 1 6-bit word format. Every program memory address contains a 16- or 32-bit instruction. During interrupts and subroutine calls, the return address

Program Counter (PC) is stored on the Stack. The Stack is effectively allocated in the general data SRAM, and consequently the stack size is only limited by the total SRAM size and the usage of the SRAM. All user programs must initialize the SP in the reset routine (before subroutines or interrupts are executed). The 8-bit Stack Pointer (SP) is read/write accessible in the I/O space. The 128 bytes data SRAM + Register File and I/O Registers can be easily accessed through the five different addressing modes supported in the AVR architecture. The memory spaces in the AVR architecture are all linear and regular memory maps.

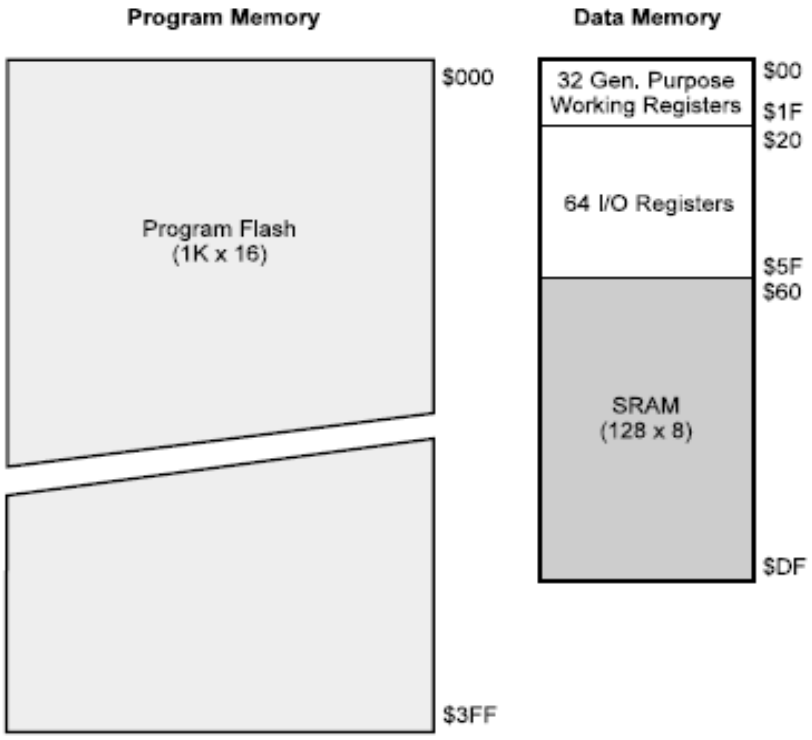


Figure 12. Memory Maps

A flexible interrupt module has its control registers in the I/O space with an additional Global Interrupt Enable bit in the Status Register. All the different

interrupts have a separate Interrupt Vector in the Interrupt Vector table at the beginning of the program memory. The different interrupts have priority in accordance with their Interrupt Vector position. The lower the Interrupt Vector address, the higher the priority.

2.8 General Purpose Register Files

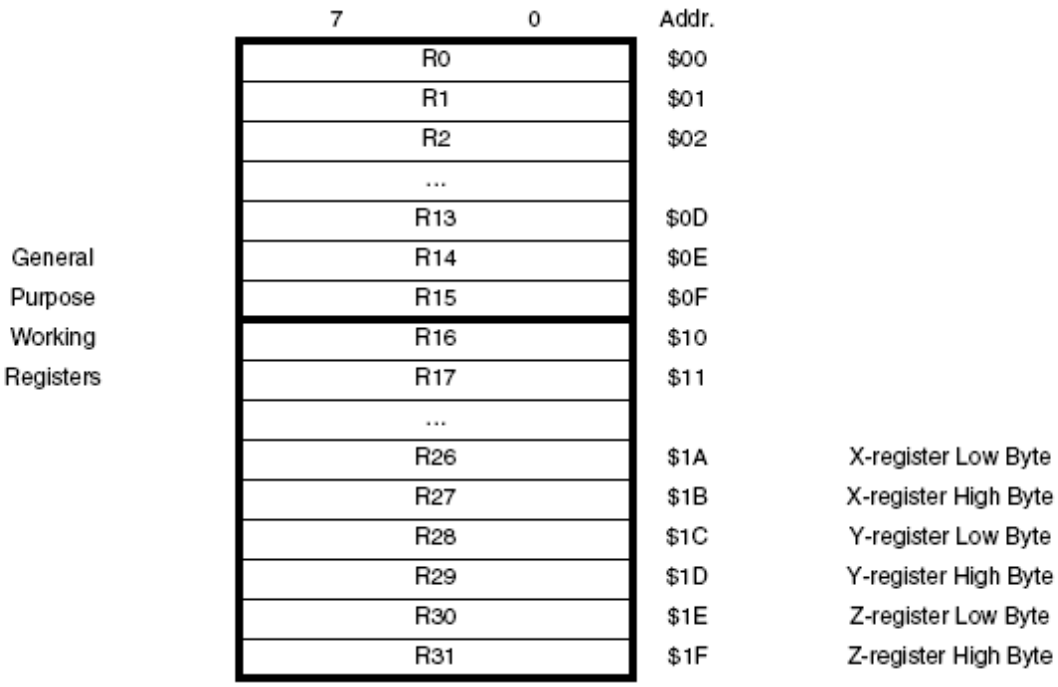


Figure 13 AVR CPU General Purpose Working Registers

All the register operating instructions in the instruction set have direct and single-cycle access to all registers. The only exception is the five constant arithmetic and logic instructions SBCI, SUBI, CPI, ANDI, ORI between a constant and a register and the LDI instruction for load immediate constant data. These instructions apply to the second half of the registers in the Register File (R16..R31). The general SBC, SUB, CP, AND, OR, and all other operations between two registers or on a single register apply to the entire Register File. As shown in Figure 6, each register is also assigned a data memory address, mapping them directly into the first 32 locations of the user Data Space. Although the Register File is not physically implemented as SRAM locations, this memory organization provides great flexibility in access of the registers, as the X-, Y-, and Z-registers can be set to index any register in the file.

2.8.1 X-register, Y-register, and Z-register

The registers R26..R31 have some added functions to their general purpose usage.These registers are the address pointers for indirect addressing of the Data Space.

The three indirect address registers X, Y and Z are defined in Figure 7.

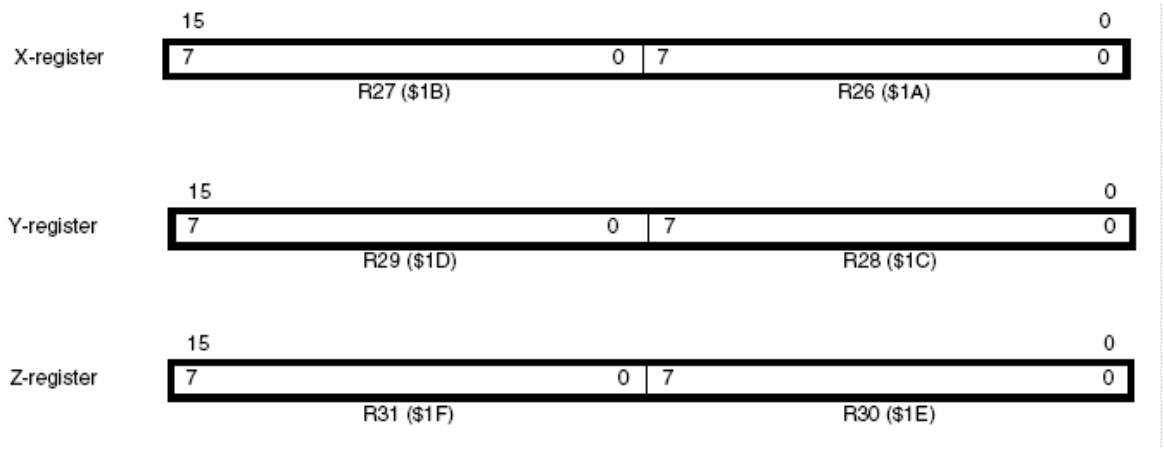


Figure 14. X-, Y-, and Z-Registers

In the different addressing modes these address registers have functions as fixed displacement, automatic increment and decrement

2.8.2 ALU - Arithmetic Logic Unit

The high-performance AVR ALU operates in direct connection with all the 32 general purpose working registers. Within a single clock cycle, ALU operations between registers in the Register File are executed. The ALU operations are divided into three main categories - arithmetic, logical, and bit functions.

2.8.3 In-System Programmable Flash Program Memory

The AT90S2313 contains 2K bytes On-chip In-System Programmable Flash memory for program storage. Since all instructions are 16- or 32-bit words, the Flash is organized as 1K x 16. The Flash memory has an endurance of at least 1,000 write/erase cycles.The AT90S2313 Program Counter (PC) is 10 bits wide, thus addressing the 1,024 program memory addresses.

2.8.4 EEPROM Data Memory

The AT90S2313 contains 128 bytes of EEPROM data memory. It is organized as a separate data space in which single bytes can be read and written. The EEPROM has an endurance of at least 100,000 write/erase cycles. The access between the EEPROM and the CPU is described on page 39, specifying the EEPROM Address Register, the EEPROM Data Register and the EEPROM Control Register.

2.4 Serial Communication

2.4.1 Serial Communication General Concept

Serial commuication is very common for devices communication which is standard on every PC , Electornic data communication etwwen elemants will generally fall into two broad categories single ended and deffrential RS 232 (single ended) was introduced in 1962 and despite rumors for its early demise has remained widely used through the industry.

Independent channels are established for two-way (full-duplex) communications. The RS232 signals are represented by voltage levels with respect to a system common (power / logic ground). The "idle" state (MARK) has the signal level negative with respect to common, and the "active" state (SPACE) has the signal level positive with respect to common. RS232 has numerous handshaking lines (primarily used with modems), and also specifies a communications protocol.

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

RS232 data is bi-polar.... +3 TO +12 volts indicates an "ON or 0-state (SPACE) condition" while A -3 to -12 volts indicates an "OFF" 1-state (MARK) condition.... Modern computer equipment ignores the negative level and accepts a zero voltage level as the "OFF" state. In fact, the "ON" state may be achieved with lesser positive potential. This means circuits powered by 5 VDC are capable of driving RS232 circuits directly, however, the overall range that the RS232 signal may be transmitted/received may be dramatically reduced.

The output signal level usually swings between +12V and -12V. The "dead area" between +3v and -3v is designed to absorb line noise. In the various RS-232-like definitions this dead area may vary. For instance, the definition for V.10 has a dead area from +0.3v to -0.3v. Many receivers designed for RS-232 are sensitive to differentials of 1v or less.This can cause problems when using pin powered widgets - line drivers, converters, modems etc. These type of units need enough voltage & current to power them self's up. Typical URART (the RS-232 I/O chip) allows up to 50ma per output pin - so if the device needs 70ma to run we would need to use at least 2 pins for power. Some devices are very efficient and only require one pin (some times the Transmit or DTR pin) to be high - in the "SPACE" state while idle. An RS-232 port can supply only limited power to another device. The number of output lines, the type of interface driver IC, and the state of the output lines are important considerations.

The types of driver ICs used in serial ports can be divided into three general categories:

- Drivers which require plus (+) and minus (-) voltage power supplies such as the 1488 series of interface integrated circuits. (Most desktop and tower PCs use this type of driver.)
- Low power drivers which require one +5 volt power supply. This type of driver has an internal charge pump for voltage conversion. (Many industrial microprocessor controls use this type of driver.)
- Low voltage (3.3 v) and low power drivers which meet the EIA-562 Standard. (Used on notebooks and laptops.)

Data is transmitted and received on pins 2 and 3 respectively. Data Set Ready (DSR) is an indication from the Data Set (i.e., the modem or DSU/CSU) that it is on. Similarly, DTR indicates to the Data Set that the DTE is on. Data Carrier Detect (DCD) indicates that a good carrier is being received from the remote modem.Pins 4 RTS (Request To Send - from the transmitting computer) and 5 CTS (Clear To Send - from the Data set) are used to control. In most Asynchronous situations, RTS and CTS are constantly on throughout the communication session. However where the DTE is connected to a multipoint line, RTS is used to turn carrier on the modem on and off. On a multipoint line, it's imperative that only one station is transmitting at a time (because they share the return phone pair). When a station wants to transmit, it

raises RTS. The modem turns on carrier, typically waits a few milliseconds for carrier to stabilize, and then raises CTS. The DTE transmits when it sees CTS up. When the station has finished its transmission, it drops RTS and the modem drops CTS and carrier together.Clock signals (pins 15, 17, & 24) are only used for synchronous communications. The modem or DSU extracts the clock from the data stream and provides a steady clock signal to the DTE. Note that the transmit and receive clock signals do not have to be the same, or even at the same baud rate.

Note: Transmit and receive leads (2 or 3) can be reversed depending on the use of the equipment - DCE Data Communications Equipment or a DTE Data Terminal Equipment.

3. GSM (GLOBAL SYSTEM FOR MOBILE COMMUNICATIONS)

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

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

- Good subjective speech quality
- Low terminal and service cost
- Support for international roaming
- Ability to support handheld terminals
- Support for range of new services and facilities
- Spectral efficiency
- ISDN compatibility

In 1989, **GSM** responsibility was transferred to the European Telecommunication Standards Institute (ETSI), and phase I of the GSM specifications were published in 1990. Commercial service was started in mid-1991, and by 1993 there were 36 GSM networks in 22 countries [6]. Although standardized in Europe, GSM is not only a European standard. Over 200 GSM networks (including DCS1800 and PCS1900) are operational in 110 countries around the world. In the beginning of 1994, there were 1.3 million subscribers worldwide [18], which had grown to more than 55 million by October 1997. With North America making a delayed entry into the GSM field with a derivative of GSM called PCS1900, GSM systems exist on every continent, and the acronym GSM now aptly stands for Global System for Mobile communications.

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

3.1 GSM Modem Unit

3.1.1. General GSM Systems Information

GSM (Global System for Mobile communication) is a technology which is the leading cell phone standard all over the world. In 1982 it was recognized as a standard for digital wireless communications and was first adopted in Europe and then by Asia, Africa etc. The first system was online in 1991 and GSM was formerly known as Group Special Mobile but now stands for Global System for Mobile communications. USA, however has not adopted GSM as a standard and so different carriers now use different technologies as opposed to only GSM.

GSM uses digital technology and the methods of time division multiple access transmission. In GSM, sound is digitally prearranged through a special encoder, which imitates the distinctiveness of human speech. This method of transmission allows a very competent statistics speed/information content ratio.

GSM is an open system and is a non propriety technology. One of the great benefits of GSM is that it facilitates international roaming. As it is adopted by more than 170 countries, you have the facility of using your GSM cell phone in all these places without having to change your number. GSM satellite roaming has broadened the scope of cellular services even to areas where standard terrestrial services are not possible.

GSM is a technology that is rapidly growing and constantly evolving with wireless, satellite and cordless systems offering greatly expanded services. These services include multimedia data services, high speed, inbuilt support for side by side

use of these services and faultless incorporation with the Internet and wireline networks. 3GSM (next generation of mobile communications services) is already charted out and will make available services enhancing the already existing voice, data, and text services. GSM will provide video on demand and will help to lessen the gap between wireless and internet/computers. GSM works on different frequency bands across the world. In North America it uses a 1900 MHz frequency whereas in other parts of the world it uses either 900MHz or 1800 MHz. As different frequencies are used in different places, your GSM handset should support various bands so that it can be used globally.

From the beginning, GSM has been developed with the need to give its customers utmost security in terms of secure communications, fraud prevention, and call privacy. Today it is the worlds most secure public wireless standard for cellular phones.

3.1.2 SMS (Short Messaging Service) for GMS Systems

SMS stands for Short Message Service. It is a technology that enables the sending and receiving of messages between mobile phones. SMS first appeared in Europe in 1992. It was included in the GSM (Global System for Mobile Communications) standards right at the beginning. Later it was ported to wireless technologies like CDMA and TDMA. The GSM and SMS standards were originally developed by ETSI. ETSI is the abbreviation for European Telecommunications Standards Institute. Now the 3GPP (Third Generation Partnership Project) is responsible for the development and maintenance of the GSM and SMS standards

As suggested by the name "Short Message Service", the data that can be held by an SMS message is very limited. One SMS message can contain at most 140 bytes (1120 bits) of data, so one SMS message can contain up to:

- 160 characters if 7-bit character encoding is used. (7-bit character encoding is suitable for encoding Latin characters like English alphabets.)
- 70 characters if 16-bit Unicode UCS2 character encoding is used. (SMS text messages containing non-Latin characters like Chinese characters should use 16-bit character encoding.)

SMS text messaging supports languages internationally. It works fine with all languages supported by Unicode, including Arabic, Chinese, Japanese and Korean.

Besides text, SMS messages can also carry binary data. It is possible to send ringtones, pictures, operator logos, wallpapers, animations, business cards (e.g. VCards) and WAP configurations to a mobile phone with SMS messages.

One major advantage of SMS is that it is supported by 100% GSM mobile phones. Almost all subscription plans provided by wireless carriers include inexpensive SMS messaging service. Unlike SMS, mobile technologies such as WAP and mobile Java are not supported on many old mobile phone models.

3.2 Used GSM (Global System for Mobile Communication) Modem of Thesis

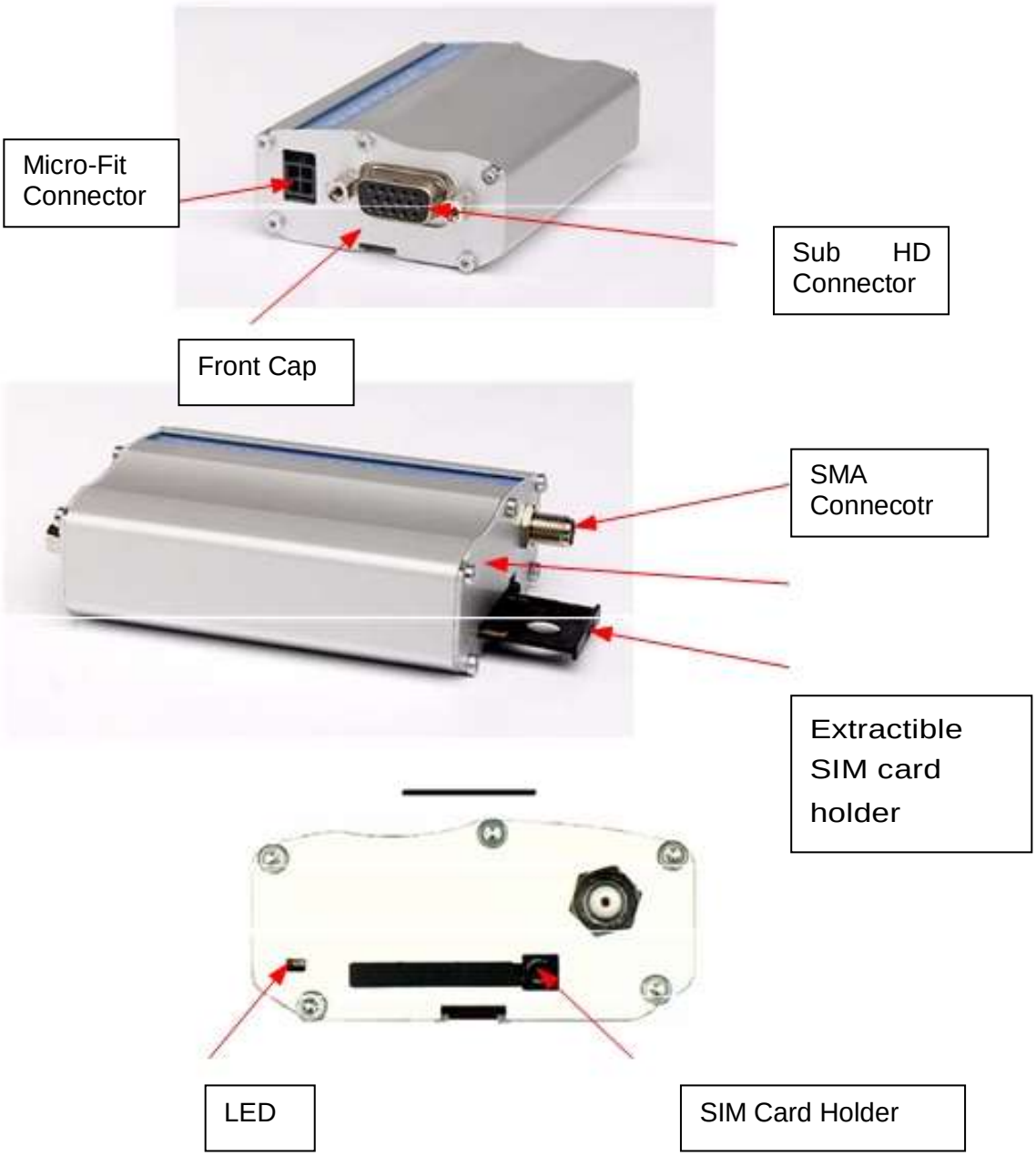
3.2.1 Wavecom Fastrack Modem M13 Series

The Fastrack M1306B is a discrete, rugged cellular modem offering state-of-theart GSM/GPRS connectivity to machine to machine applications.

Proven for reliable, stable performance on wireless networks worldwide, WAVECOM's latest generation of Fastracks -M1306B- continues to deliver rapid time to market and painless integration.

Smaller than the former generation and updated with new features, the M1306B now offers 2 general purpose input/output access.

Fully certified, the dual band 900/1800 MHz Fastrack M1306B offers GPRS cl.10 capability and supports a powerful open software platform (Open AT) enabling cost-efficient designs to be achieved and capable of hosting any industrial or IT protocol. Fastrack M1306B is controlled by firmware through a set of AT commands


Micro-Fit Connector
Sub HD Connector
Front Cap
SMA Connecotr
Extractible SIM card holder
LED
SIM Card Holder

3.2.2 Basic Features and Services

3.2.3 Connectivity Control and Main AT Command of GSM Modem

Features	**GSM**	**DCS**
Standart	900 MHz. E-GSM compliant. Output power: class 4 (2W). Fully compliant with ETSI GSM phase 2 + small MS	1800 MHz Output power: class 1 (1W). Fully compliant with ETSI GSM phase 2 + small MS.
GPRS	Class 10. PBCCH support. Coding schemes: CS1 to CS4.CompliantwithSMG31bi Embedded TCP/IP stack (optional).	
INTERFACES	RS232 (V.24/V.28) Serial interface supporting: Baud rat(bits/s):300,600,1200,2400, 4800, 9600, 19200, 38400, 57600, 115200, Autobauding (bits/s):2400, 4800, 9600, 19200, 38400, 57600. 2 General PurposeInput/Output gates (GPIOs) available. 3 V SIM interface. AT command set based on V.25ter and GSM 07.05 & 07.07. OpenAT interface for embedded application.	
SMS	Text & PDU. Point to point (MT/MO). Cell broadcast.	
GPRS	Class 10. PBCCH support. Coding schemes: CS1 to CS4.CompliantwithSMG31bi Embedded TCP/IP stack (optional).	

Using a communication software such as Hyperterminal, enter the AT command. The response of the modem must be OK displayed in the Hyperterminal window. If the communication cannot be established with the modem, do the following:

- Check the RS232 connection between the DTE and the modem (DCE),
- Check the configuration of the port COM used on the DTE.

Example of AT commands which can be used after getting started the modem:

- AT+CGMI: modem answer is “WAVECOM MODEM” when serial link is OK.
- AT+CPIN=xxxx: to enter a PIN code xxxx (if activated). AT+CSQ: to verify the received signal strength.
- AT+CREG?: to verify the registration of the modem on the network.
- ATD<phone number>;: to initiate a voice call. ATH: to hang up (end of call).

AT commands are instructions used to control a modem. AT is the abbreviation of ATtention. Every command line starts with "AT" or "at". That's why modem commands are called AT commands. Many of the commands that are used to control wired dial-up modems, such as ATD (Dial), ATA (Answer), ATH (Hook control) and ATO (Return to online data state), are also supported by GSM/GPRS modems and mobile phones. Besides this common AT command set, GSM/GPRS modems and mobile phones support an AT command set that is specific to the GSM technology, which includes SMS-related commands like AT+CMGS (Send SMS message), AT+CMSS (Send SMS message from storage), AT+CMGL (List SMS messages) and AT+CMGR (Read SMS messages).

Note that the starting "AT" is the prefix that informs the modem about the start of a command line. It is not part of the AT command name. For example, D is the actual AT command name in ATD and +CMGS is the actual AT command name in AT+CMGS. However, some books and web sites use them interchangeably as the name of an AT command.

Here are some of the tasks that can be done using AT commands with a GSM/GPRS modem or mobile phone:

- Get basic information about the mobile phone or GSM/GPRS modem. For example, name of manufacturer (AT+CGMI), model number (AT+CGMM), IMEI number (International Mobile Equipment Identity) (AT+CGSN) and software version (AT+CGMR).
- Get basic information about the subscriber. For example, MSISDN (AT+CNUM) and IMSI number (International Mobile Subscriber Identity) (AT+CIMI).
- Get the current status of the mobile phone or GSM/GPRS modem. For example, mobile phone activity status (AT+CPAS), mobile network registration status (AT+CREG), radio signal strength (AT+CSQ), battery charge level and battery charging status (AT+CBC).
- Establish a data connection or voice connection to a remote modem (ATD, ATA, etc).
- Send and receive fax (ATD, ATA, AT+F*).
- Send (AT+CMGS, AT+CMSS), read (AT+CMGR, AT+CMGL), write (AT+CMGW) or delete (AT+CMGD) SMS messages and obtain notifications of newly received SMS messages (AT+CNMI).
- Read (AT+CPBR), write (AT+CPBW) or search (AT+CPBF) phonebook entries.
- Perform security-related tasks, such as opening or closing facility locks (AT+CLCK), checking whether a facility is locked (AT+CLCK) and changing passwords (AT+CPWD).

(Facility lock examples: SIM lock [a password must be given to the SIM card every time the mobile phone is switched on] and PH-SIM lock [a certain SIM card is associated with the mobile phone. To use other SIM cards with the mobile phone, a password must be entered.])

- Control the presentation of result codes / error messages of AT commands. For example, you can control whether to enable certain error messages (AT+CMEE) and whether error messages should be displayed in numeric format or verbose format (AT+CMEE=1 or AT+CMEE=2).
- Get or change the configurations of the mobile phone or GSM/GPRS modem. For example, change the GSM network (AT+COPS), bearer service type (AT+CBST), radio link protocol parameters (AT+CRLP), SMS center address (AT+CSCA) and storage of SMS messages (AT+CPMS).

- Save and restore configurations of the mobile phone or GSM/GPRS modem. For example, save (AT+CSAS) and restore (AT+CRES) settings related to SMS messaging such as the SMS center address.

Note that mobile phone manufacturers usually do not implement all AT commands, command parameters and parameter values in their mobile phones. Also, the behavior of the implemented AT commands may be different from that defined in the standard. In general, GSM/GPRS modems designed for wireless applications have better support of AT commands than ordinary mobile phones.

In addition, some AT commands require the support of mobile network operators. For example, SMS over GPRS can be enabled on some GPRS mobile phones and GPRS modems with the +CGSMS command (command name in text: Select Service for MO SMS Messages). But if the mobile network operator does not support the transmission of SMS over GPRS, you cannot use this feature.

3.2.4 Functional Description

3.2.4.1 Architecture

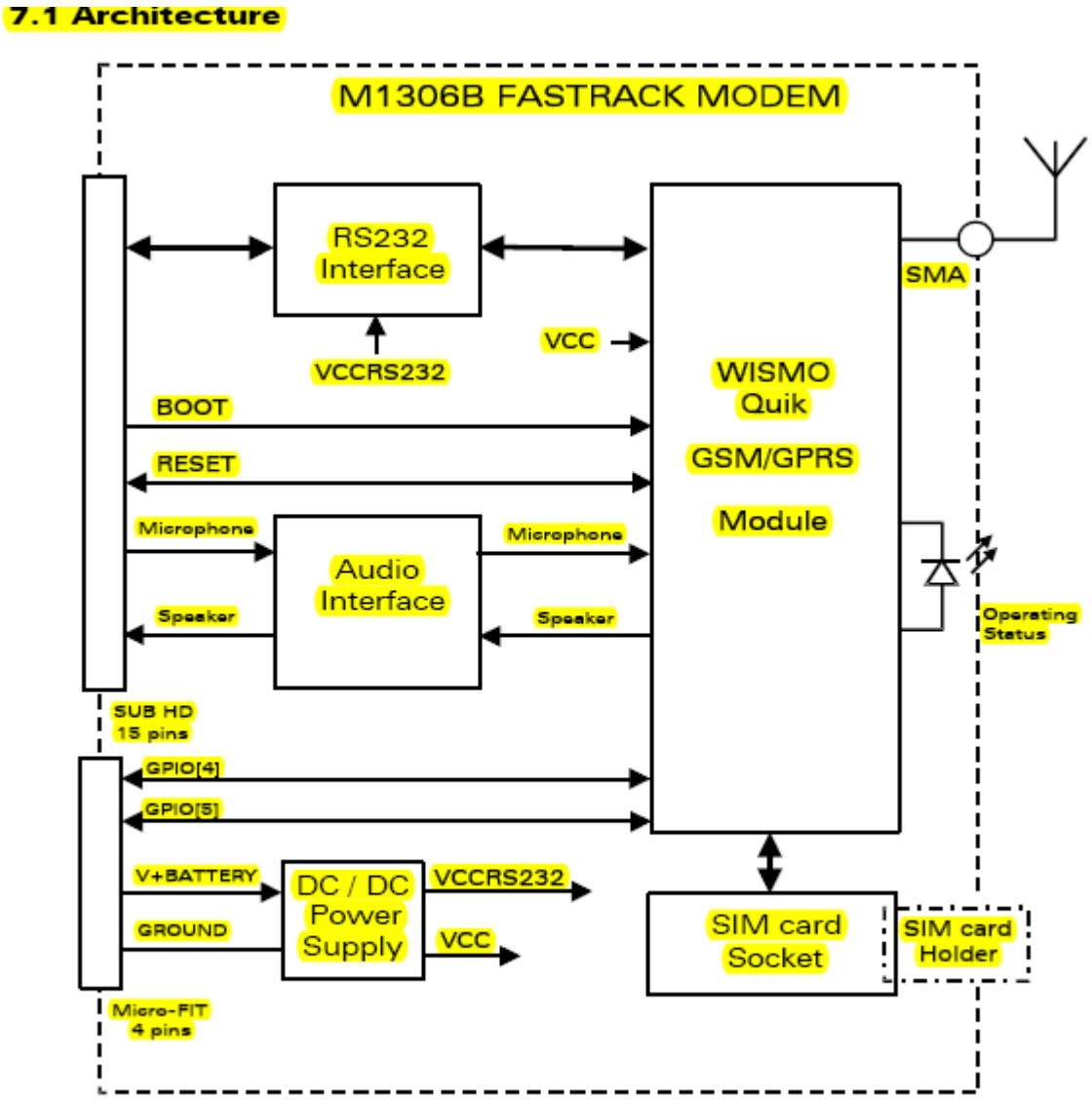


Figure 15 Functional Architecture of Wavecom Fastrack

3.2.5 RS 232 Serial Link

The RS232 interface performs the voltage level adaptation (V24/CMOS. ⇔ V24/V28) between the internal WISMO module (DCE) and the external world (DTE). RS232 interface is internally protected electrostatic surges on the RS232 lines.

Filtering guarantees: EMI/RFI protection in input and output, Signal smoothing.

Signals available on the RS232 serial link are:

TX data (CT103/TX),

RX data (CT104/RX),

Request To Send (CT105/RTS),

Clear To Send (CT106/CTS),

Data Terminal Ready (CT108-2/DTR),

Data Set Ready (CT107/DSR),

Data Carrier Detect (CT109/DCD),

Ring Indicator (CT125/RI).

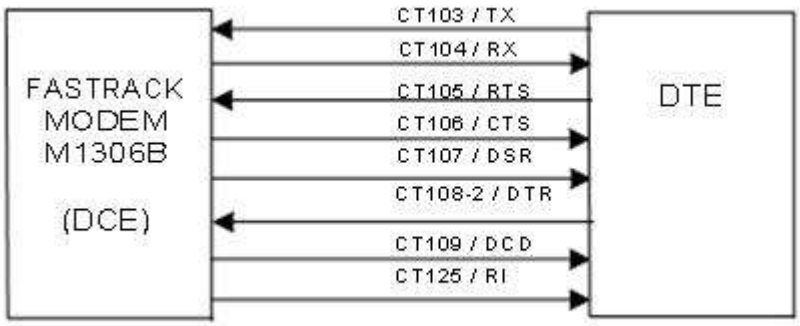


Figure 16 RS 232 Connections

RS232 interface has been designed to allow a certain flexibility in the use of the serial interface signals. However, the use of TX, RX, CTS and RTS signals is mandatory which is not the case for DTR, DSR, DCD and RI signals which can be not used.

3.3 Wismo Quick 2406B Series GSM Modul

3.3.1 General Specification

WISMO Quik Q24x6 sub-series is a range of self-contained E-GSM/GSM-GPRS 900/1800 or 850/1900 dual-band modules including the following features:

-58.4 x 32.2 x 3.9 mm.

-2 Watts E-GSM 900/GSM 850 radio section running under 3.6 Volts.

-1 Watt GSM1800/1900 radio section running under 3.6 Volts.

- Digital section running under 2.8 Volts.

-3V only SIM interface (for both 1.8 and 5 V SIM interface with external adaptation, refer to document [1]).

-Real Time Clock with calendar.

-Battery charge management.

-Echo Cancellation + noise reduction.

-Full GSM or GSM/GPRS software stack.

-Hardware GPRS class 10 capable.

-Complete shielding.

-Complete interfacing through a 60-pin connector:
-Power supply,

-Serial link,
-Audio,

-SIM card interface,
-Keyboard,

-LCD (not available with AT commands).

WISMO Quik Q24x6 sub-series has two external connections:

-RF connection pads (to the antenna),

-60-pin General Purpose Connector (GPC) to Digital, Keyboard, Audio and Supply.

WISMO Quik Q24x6 sub-series is designed to fit in very small terminals and only some custom functions have to be added to make a complete dual-band solution:

-Keypad and LCD module,

-Earpiece and Microphone,

-Base connector,

-Battery,

-Antenna,

-SIM connector.

3.3.2 Functional Description

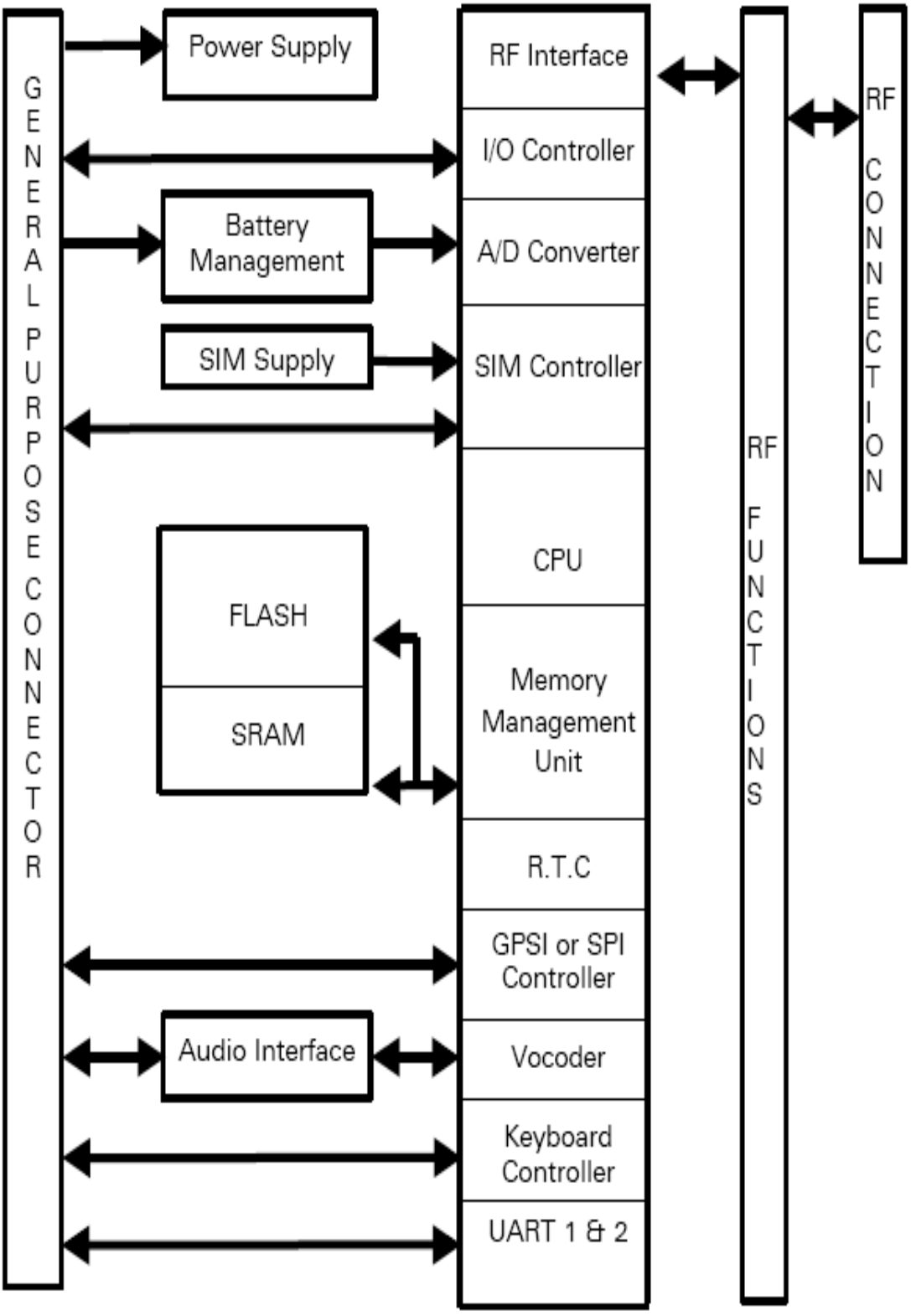


Figure 17 Functional Description Vismo GSM Module

3.3.3 RF Baseband Functionalities

The Radio Frequency (RF) functionalities comply with the Phase II E-GSM 900/DCS 1800 and GSM 850/PCS 1900 recommendation.

The frequencies are: Q2406

Rx (E-GSM 900): 925 to 960 MHz.
Rx (DCS 1800): 1805 to 1880 MHz.
Tx (E-GSM 900): 880 to 915 MHz.
Tx (DCS 1800): 1710 to 1785 MHz. Q2426
Rx (GSM 850): 869 to 894 MHz.
Rx (PCS 1900): 1930 to 1990 MHz.
Tx (GSM 850): 824 to 849 MHz.
Tx (PCS 1900): 1850 to 1910 MHz.

The Radio Frequency (RF) part is based on a specific dual band chip including : Low-IF Receiver, Dual RF (Radio Frequency) synthesizer, Digital IF to Baseband Converter, Offset PLL (Phase Lock Loop) transmitter, 1 (logarithmic) Power Amplifier (PA) controller, Dual band Power Amplifier (PA) module.

Baseband functionalities :The digital part of the WISMO Quik Q24x6 sub-series is based on a PHILIPSVLSI chip (ONE C GSM/GPRS Kernel). This chipset is using a 0.25 µm mixed technology CMOS, which allows massive integration as well as low current consumption

4. OPEN AT SOFTWARE DEVELOPMENT

4.1 Software Architecture

The Application Development Layer software library, based on standard embedded Open AT API layer, is included in the Wavecom library since Open AT release 2.00 (as defined in section 2.1.1 ”Software Organization” of the Basic Development Guide).

The aim of the ADL is to provide a high level interface to the Open AT software developer. The ADL supplies the mandatory software skeleton for an embedded application, for instance the message parser (see 2.2: “Minimum Embedded Application Code” of Open AT Basic Development Guide) and some messages states machines for given complex services (SIM service, SMS service…).

Thus, the Open AT software developer can concentrate on the contents of his application. He or she simply has to write the callback functions associated to each service he or she wants to use.

Therefore the software supplied by Wavecom contains the items listed below:

ADL software library wmadl.lib,
A set of header files (.h) defining the ADL API functions,
Source code samples,
It relies on the following software architecture:

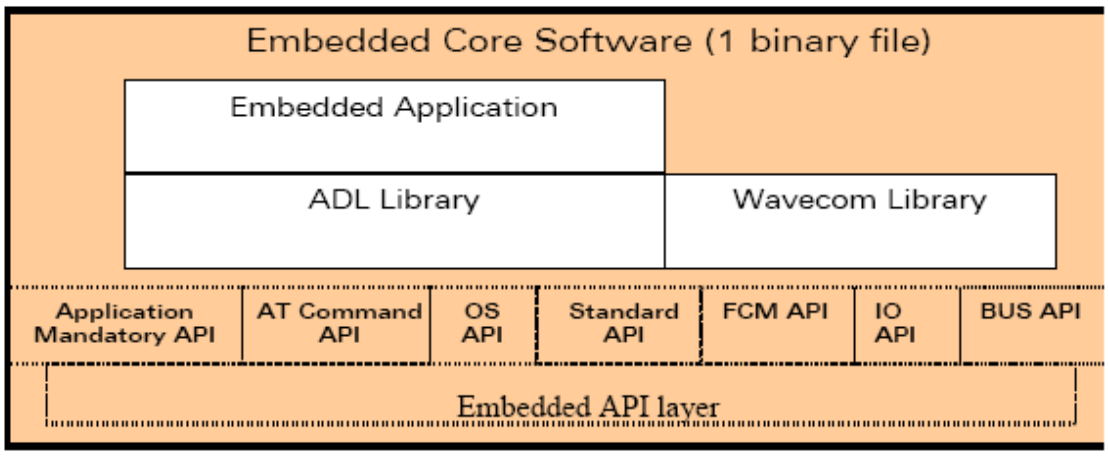


Figure 18 Software Architecture

4.1.1 Minimum Embedded Application Code

The following code must be included in any Embedded Application:

```
u32 wm_apmCustomStack [ 256 ];

/* the value 256 is an example */

const u16 wm_apmCustomStackSize = sizeof (wm_apmCustomStack;

s32 wm_apmAppliInit (wm_apmInitType_e   InitType)
{

return OK;

}

s32 wm_apmAppliParser ( wm_apmMsg_t * Message )

{

return OK;

}
```

Important Remark : in former Open-AT versions, the wm_apmCustomStack variable was declared as an u8 table. This is modified since version 2, when wm_apmCustomStack became an u32 table, for memory alignment compatibility purpose with ADS compiler.

4.1.2 Specificity of AT Commonds in The Open AT Architecture

4.1.2.1 AT Command Size

The maximum size of an AT command string or a Response string that can be sent through the serial link is 512 bytes. Therefore, if the Embedded Application needs to send more data, it must be sent in several increments.

4.1.2.2 AT+WDWL Command

By default the AT+WDWL command, used to download an application, is not pre-parsed. Therefore, even if the Embedded Application has subscribed to the command pre-parsing mechanism, this command is processed by means of the Wavecom software and it is not sent back to this application.
The embedded application may also pre-parse it to prevent external application downloading another application over it.

4.1.2.3 AT+WOPEN Command

Open-AT require some specific AT commands such as AT+WOPEN. The latter is described below. By default this command is always available for an External Application. It is not pre-parsed and it is processed even if the AT software is busy.

The embedded application may also pre-parse it to prevent external application stopping it. This command deactivates an Embedded Application in order to ensure that a new application can be downloaded. Typically, if an Embedded Application continuously sends AT commands, the Wavecom AT command software is always busy. Therefore, if the AT+WDWL command is sent by an External Application, it is not processed.

AT+WOPEN can take the values

0 Stop,

1 Start,

2 Get the Open AT library versions,

3 Erase the objects flash of the Open-AT Embedded Application,

4 Erase the Open-AT Embedded Application).

5 Suspends the Open-AT application tasks (same effect than the

wm_osSuspend API). Open-AT application running in Remote Task Environment can not be suspended (the command has no effect).

Sending the AT+WOPEN=0 command first, by means of an External Application, deactivates the Embedded Application: a new Embedded Application may then be downloaded, or the actual objects flash can be erased or the actual Open-AT Embedded Application can be erased.

If the Embedded Application is deactivated, it can be restarted using

AT+WOPEN=1. The module then reboots and this application is restarted 20 sec

after the module boot. In this case, the objects flash or the application can't be erased

You can check if the OpenAT feature is enabled with the command AT+WOPEN=? . In the case of feature disabled, the answer is +CME ERROR 3. Else it is +WOPEN: (0-4).

4.2 Memory Management

The Embedded software runs within an RTK task: the user must define the size of the customer application call stack.

The Wavecom Core Software and the Embedded Application manage their own RAM area. Any access from one of these programs to the other's RAM area is prohibited and causes a reboot.

In case an Embedded Application uses more than the maximum allocated RAM in global variables, or uses more than the maximum allocated ROM, then the behavior of the Embedded software becomes erratic.

Global variables, call stack and dynamic memory are all part of the RAM allocated to the Embedded Application.

The available memory sizes are :

For 16 Mbits flash size products ('A' WISMO module series):

256 Kbytes of ROM

32 Kbytes of RAM

5 Kbytes of Flash Object Data

0 Kbytes of Application & Data Storage Volume

For 32 Mbits flash size products ('B' WISMO module series):

512 Kbytes of ROM

128 Kbytes of RAM

128 Kbytes of Flash Object Data

512 Kbytes of Application & Data Storage Volume

2.5 Known Limitations

4.3 Command Pre-Parsing Limitation

In normal operating mode “command mode”, the target serial link manager checks to see whether every command starts with “AT” and ends with a carriage return and end-of-line chars. Therefore, the only commands to be dispatched to the Embedded Application (in case of command pre-parsing subscription) are the ones complying with the here-above description.

Remark:

If you want to receive particular commands which are not AT commands

(starting with another thing than “AT”), you can use the “data mode” to send and receive these commands into the Embedded Application

4.4 Missing Unsolicited Messages in Remote Application

In Remote Application Execution mode, the application is started a few seconds after the Target. Therefore, some unsolicited events might be lost.

A pre-processor flag like __REMOTETASKS__ can be used to add some specific code for remote mode.

4.4.1 Minimum Embedded Application Code

The following code must be included in any Embedded Application:

```
const wm_apmTask_t wm_apmTask [ WM_APM_MAX_TASK ] =
{

{ StackSize1, Stack1, InitFct1, ParserFct1 },
{ StackSize2, Stack2, InitFct2, ParserFct2 },
```

{ StackSize3, Stack3, InitFct3, ParserFct3 }
};

StackX and StackSizeX are variables used to define the application tasks call stack size

InitFctX() are functions which are the first called ones for each Embedded Application task

ParserFctX() are functions which are called each time an Embedded

Application task receives a message from the Wavecom Core Software

4.5 Security

4.5.1 Software Security

Two software safeguards are used in the Open-AT platform:o RAM access protection o watchdog protection.

After reboot, the “Init ()” function of each task will have its parameter set to WM_APM_REBOOT_FROM_EXCEPTION.

After a reboot caused by a soware crash, the application is started only 20 seconds after the start of the Wavecom Core software. This allows at least 20 seconds delay to re-download a new application.

In case of normal reboot, the application restars immediately.

4.5.2 RAM Access Protection

A specific RAM area is allocated to the Embedded Application. The Embedded Application is seen as a Real-Time Task in the Wavecom software, and each time this task runs, the Wavecom RAM protection is activated. If the Embedded Application tries to access this RAM, then an exception occurs and the software reboots. In case of illegal RAM access, the Target Monitoring Tool screen displays: ”ARM exception 1 xxx,” where “xxx” is the address the application was attempting to access. If the symbol file is correctly configured in the Target Monitoring Tool, then a Back Trace must describe the affected “C” functions in which the crash occurred.

4.5.3 Watchdog Protection

The Embedded Application software is protected from reaching a dead-end lock by a 8 seconds watchdog. In case of a crash, the software reboots.

If an Embedded Application crash is detected, the Target Monitoring Tool screen displays: ”Customer watchdog.”

4.5.4 Hardware Security

Protection can also be improved using an external watchdog reset circuitry.With such a hardware watchdog protection, the Wavecom product will always be reset even in case of the software crashes. To achieve this, one can use a GPO connected to a specific hardware counter that will reset the product if not refreshed. For example, this specific hardware can be a counter with a specific counter output connected to the reset pin of the module, and the counter reset pin connected to a GPO. In this way, the software in the module is supposed to reset the counter periodically. If not, the counter will increase until it reaches the specified limit and then resets the module.

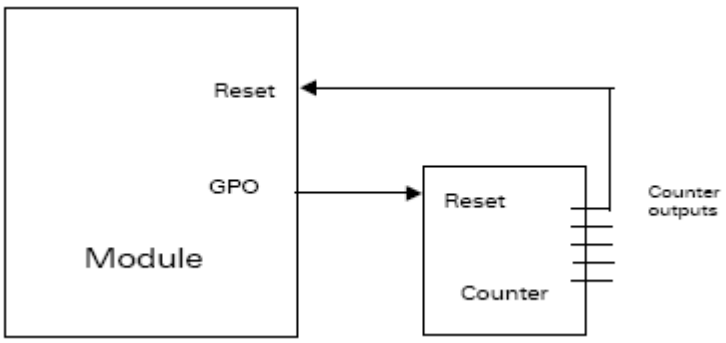


Figure 19 Resetting GSM Module

5. THESIS APPLICATION

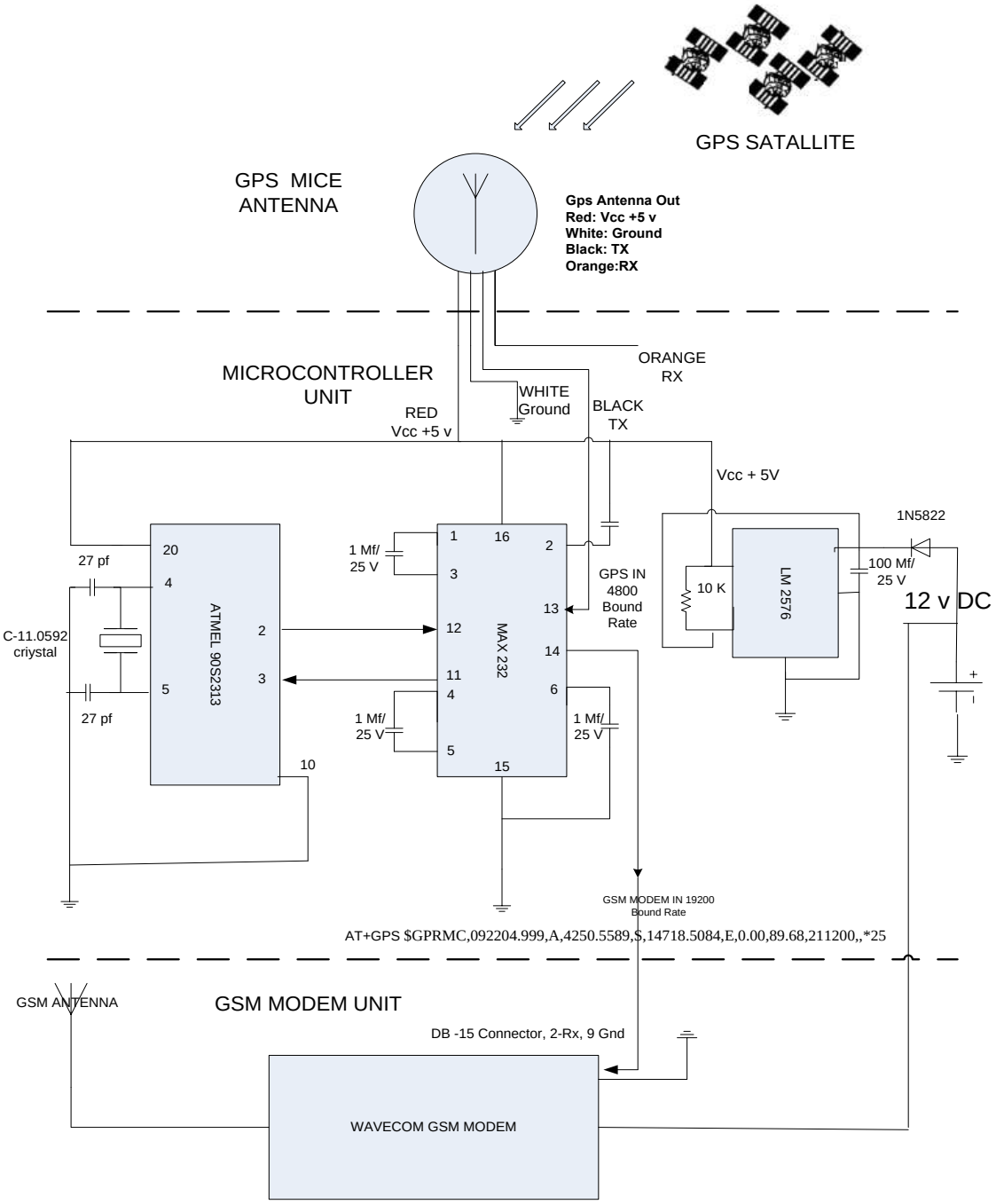


Figure 20 Designed Atmel Microcontroller Unit block diagram

At this Project, mobile unit consist of GPS mouse receiver which is receive data from GPS satellites and send NMEA protocol based signal by RS 232. Microcontroller (Atmel 90s2313) receive the GPS data and filtering valid data,and send data by using AT+GPS command ,then wavecom GSM&GPRS modem send this valid data, dedicated time range or if request last location send GPS data by using SMS to central unit. Components specification detail explained at follow;

The Designed system is a vehicle tracking system which is check the position and velocity of the vehicle by using GPS and then send the valid GPS data to a GSM telephone number by using GSM network.

AVR Atmel 90S2313 Microcontroller is used to control received of the GPS data. In order to send received valid GPS data a GSM telephone as a SMS Wavecom GSM modem is used. Main working principal of the sytem is GPS Mouse receive NMEA sentences then send the data to AVR Microcontroller

GPS Mouse Pin out is;

White : Ground

Red: Vcc

Orange: RX

Black: TX

Micro Controller Pin Out:

Number 13pin is RX

Number 14 pin is TX

Wavecom GSM Modem Pins DB 15;

Number 2 pin is RX

Number 9 pin is Ground

5.1 How to Set GPS Mouse Receiver

NemeriX Control Tool is used to set GPS Mouse which is connect 4800 Bound rate. In order to view valid GPS data select to SET NMEA button then select RMC and set the time range received data period. It is shown Figure 20.

“$GPRMC,092204.999,A,4250.5589,S,14718.5084,E,0.00,89.68,211200,,*25

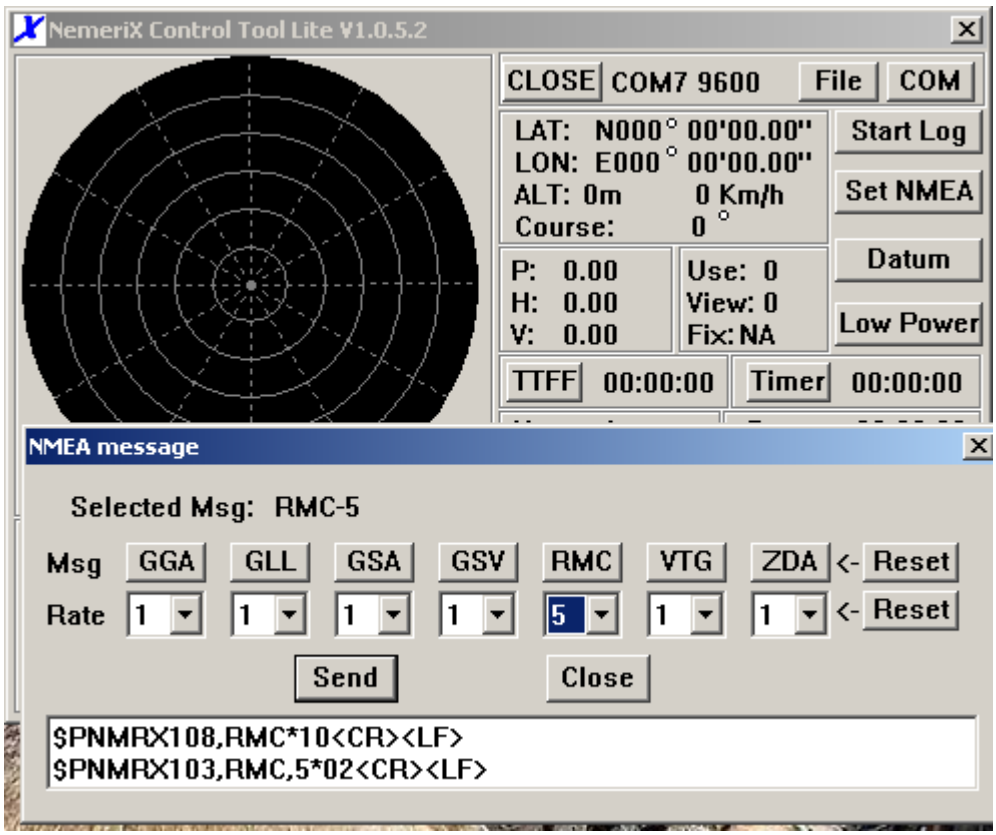


Figure 21 NemeriX GPS Control Tool

5.2 Wavecom GSM Modem Command

Wavecom GSM modem is programammed by using Open AT commands,In the software some constant is used to set Central GSM Number and set SMS sending period.

AT+MERKEZ ? used existing GSM Number,

AT+MERKEZ:+90505377778

AT+GPS? Last received GPS position information

AT+PERIOD? Used viewed SMS send period.

AT+PERIOD: 00060 Sn Used to set SMS send time period.

5.3 SOFTWARE ON ATMEL 90S2313 MICROCONTROLLER

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/eeprom.h>
#include <string.h>

#define F_CPU            11059200
//#define F_CPU                  3686400

#define CYCLES_PER_US ((F_CPU+500000)/1000000)
#define UART_TX_BAUD_RATE    19200
#define UART_RX_BAUD_RATE    4800
#define UART_INTERRUPT_HANDLER  SIGNAL

#ifdef UCSRB
    #define UCR                         UCSRB
#endif

#if defined(UBRR) && !defined(UBRRL)
    #define     UBRRL                       UBRR
#endif

#ifndef outb
    #define     outb(addr, data)    addr = (data)
#endif

#ifndef inb
    #define     inb(addr)                   (addr)
#endif

#ifndef BV
    #define BV(bit)                     (1<<(bit))
#endif

#ifndef cbi
```

```
        #define cbi(reg,bit)    reg &= ~(BV(bit))
#endif

#ifndef sbi
        #define sbi(reg,bit)    reg |= (BV(bit))
#endif

#ifndef cli
        #define cli()                   __asm__ __volatile__ ("cli" ::)
#endif

#ifndef sei
        #define sei()                   __asm__ __volatile__ ("sei" ::)
#endif

#define UART_BUFFER_SIZE      100
#define UART_MODE_TX          0x31
#define UART_MODE_RX          0x32
#define UART_MODE_TXRX              0x33
#define EEPROM_SIZE           128
#define GPS_STATE_HEADER      0x01
#define GPS_STATE_DATA        0x02
#define GPS_STATE_LF          0x03
#define UART_BUFFER_OFFSET    0x08

typedef unsigned char  u08;
typedef unsigned int   u16;
typedef unsigned long  u32;


void uartInit(u08 mode);
void uartSetBaudRate(u32 baudrate);


u08     uartReadyTx;
u08     GPS_rx_state;
u08     GPS_rx_index;
```

```
u08     uartBuffer[UART_BUFFER_SIZE];
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
void uartInit(u08 mode)
{
        cli();
        outb(UCR, 0x00);
        switch(mode) {
                case UART_MODE_TX:
                        uartSetBaudRate(UART_TX_BAUD_RATE);
                        outb(UCR, BV(TXCIE)|BV(TXEN));
                        break;

                case UART_MODE_RX:
                        uartSetBaudRate(UART_RX_BAUD_RATE);
                        outb(UCR, BV(RXCIE)|BV(RXEN));
                        break;

                case UART_MODE_TXRX:
                        uartSetBaudRate(UART_TX_BAUD_RATE);
                        outb(UCR,
BV(RXCIE)|BV(TXCIE)|BV(RXEN)|BV(TXEN));

                        break;
        }
        uartReadyTx = 1;
        sei();
}
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
void uartSetBaudRate(u32 baudrate)
{
        u16 bauddiv = ((F_CPU+(baudrate*8L))/(baudrate*16L)-1);
        outb(UBRRL, bauddiv);
#ifdef UBRRH
        outb(UBRRH, bauddiv>>8);
#endif
```

```
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
void uartSendByte(u08 txData)
{
	while(!uartReadyTx);
	outb(UDR, txData);
	uartReadyTx = 0;
}
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
void GPRS_send_data(u08 *buffer, u08 len)
{
	u08 i;
	//¦u08 *p;

	uartInit(UART_MODE_TX);

	uartSendByte('A');
	uartSendByte('T');
	uartSendByte('+');
	uartSendByte('G');
	uartSendByte('P');
	uartSendByte('S');
	uartSendByte('=');
	uartSendByte("");
	for(i=0; i<len; i++) {
		if (buffer[i]!=0x00)
			uartSendByte( buffer[i] );
		else
		break;
	}
	uartSendByte("");
	uartSendByte('\r');
	uartSendByte('\n');

	while(!uartReadyTx);
```

```
	uartInit(UART_MODE_RX);
}
//------------------------------------------------------------------------
//------------------------------------------------------------------------
UART_INTERRUPT_HANDLER(SIG_UART_TRANS)
{
	uartReadyTx = 1;
}
//------------------------------------------------------------------------
//------------------------------------------------------------------------
UART_INTERRUPT_HANDLER(SIG_UART_RECV)
{
	u08 rx_byte, i;

	rx_byte = inb(UDR);

	switch(GPS_rx_state) {
		case GPS_STATE_HEADER:
			if (rx_byte == '$') {
				GPS_rx_index=1;

				uartBuffer[0] = rx_byte;
				GPS_rx_state = GPS_STATE_DATA;
			}
			break;

		case GPS_STATE_DATA:
			uartBuffer[GPS_rx_index++] = rx_byte;
			if (rx_byte == 0x0D) {

				GPS_rx_state = GPS_STATE_LF;
			}

			break;

		case GPS_STATE_LF:
			uartBuffer[GPS_rx_index++] = rx_byte;
```

```
                        if (rx_byte == 0x0A) {

                                if (!strncmp(&uartBuffer[1], "GPRMC", 5))
                                {
                                        i = 7;

                                        while(uartBuffer[i++] != ',');

                                        if (uartBuffer[i]=='A') {
                                                memcpy(&uartBuffer[13],
&uartBuffer[17], GPS_rx_index-4);
                                                GPRS_send_data(&uartBuffer[0],
(GPS_rx_index-6) );
                                                GPS_rx_state =
GPS_STATE_HEADER;
                                        }
                                }
                                sei();

//memset(&uartBuffer[0], 0xFF, UART_BUFFER_SIZE);
        GPS_rx_state = GPS_STATE_HEADER;
                        }
                        else {
                                GPS_rx_index=0;
                                GPS_rx_state = GPS_STATE_HEADER;
                                //memset(&uartBuffer[0], 0xFF,
UART_BUFFER_SIZE);
                        }
                        break;
        }
}

//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
int main(void)
{

        GPS_rx_index = 0;
        GPS_rx_state = GPS_STATE_HEADER;
```

```
        memset(&uartBuffer[0], 0x00, UART_BUFFER_SIZE);
        uartInit(UART_MODE_RX);
                sei();
        do {
        }while(1);
}
```

5.4 PROGRAMMABLE GSM MODULE

This Module is programmable by using Open AT command and C compiler. Wismo Q2400

```
/*****************************************************************************
*************************/
/*  fatih.c   -  Copyright SyS (c) 2005
                                        */
/*THESIS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER ENGINEERING PROGRAMME */
                                        */

/*****************************************************************************
*************************/

/*****************************************************************************
********/
/*  File       :fatih.c                                    */
/*---------------------------------------------------------------------------*/
/*  Object     : Fatih Erturk Project                                  */
/*                                                          */
/*  School          : Haliç  University                             */
/*                                                          */
/*  subject    : Microcontroller Based Tracking systems
*/
/*****************************************************************************
********/
#include "adl_global.h"


const ascii* _AT_PERIOD_CMD="AT+PERIOD";
const ascii* _AT_MERKEZ_CMD="AT+MERKEZ";
const ascii* _AT_GPS_CMD="AT+GPS";

const ascii HND[4]="HND\0";
```

```
#define ATCMDTYPE
        ADL_CMD_TYPE_TEST|ADL_CMD_TYPE_READ|ADL_CMD_TYPE_P
ARA|0X0011

#define AT_PERIOD_CMD (ascii*) _AT_PERIOD_CMD
#define AT_MERKEZ_CMD (ascii*) _AT_MERKEZ_CMD
#define AT_GPS_CMD (ascii*) _AT_GPS_CMD

#define ATRSP(_X_)          adl_atSendResponse(ADL_AT_RSP, _X_);

typedef enum
{
        emod_Nop,
        emod_SendSms,
        emod_null
}EMOD;
/*************************************************************************
********/
/*  Mandatory variables                                    */
/*-------------------------------------------------------------------------*/
/*  wm_apmCustomStack                                        */
/*  wm_apmCustomStackSize                                     */
/*-------------------------------------------------------------------------*/
/*************************************************************************
********/
u32 wm_apmCustomStack [ 256 ];
const u16 wm_apmCustomStackSize = sizeof ( wm_apmCustomStack );
EMOD emod=emod_Nop;
s8 Status;
u8 SmsHandle;

static u16 iPeriod=0x0001;
static u16 iMERKEZ=0x0002;

ascii fPeriod[6];
ascii fMERKEZ[16];
ascii fGPS[200];
```

```
u16 SendTime=1;

void timerHandler(u8);
ascii* parse_string(ascii *Buffer, u8 StartChar, u8 EndChar);


void SMS_Control_Handler(u8 Event, u16 Nb)
{

}

bool SMS_Handler (ascii * SmsTel, ascii* SmsTimeLength, ascii* SmsText)
{
	return TRUE;
}

void timerHandler1(u8 ID)
{
	adl_tmrSubscribe(TRUE,10,ADL_TMR_TYPE_100MS,timerHandler);
}
void timerHandler(u8 IDD)
{
	switch(emod)
	{
			case emod_Nop:
				SendTime++;
				if (SendTime == wm_atoi(fPeriod))
				{
					emod=emod_SendSms;
					SendTime=1;
				}
			break;

			case emod_SendSms:
				if (wm_strlen(fMERKEZ) > 5)
				{
```

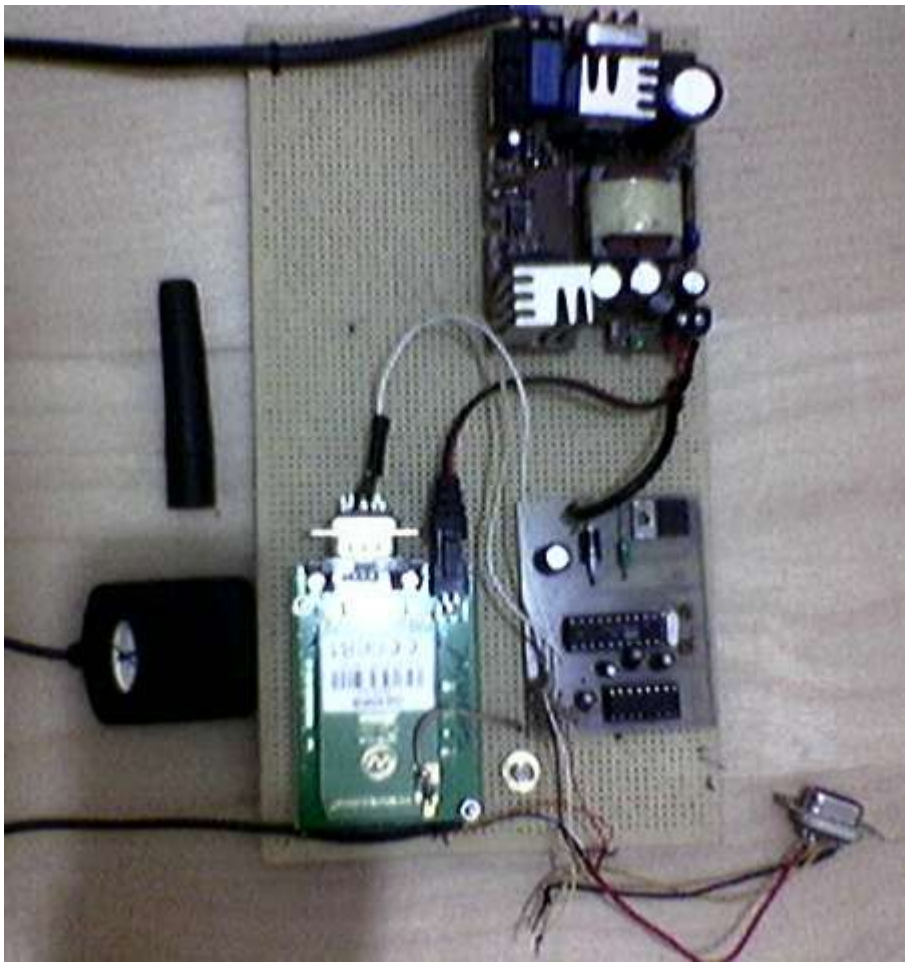
```
                                        adl_smsSend(SmsHandle, (ascii *)fMERKEZ,
(ascii *) fGPS, ADL_SMS_MODE_TEXT);
                                }
                                emod=emod_Nop;
                        break;


        }
}



void AT_PERIOD_CMD_Handler(adl_atCmdPreParser_t *paras)
{
        ascii *param0, RspStr[30];

        switch (paras->Type)
        {
                case ADL_CMD_TYPE_TEST:
                        wm_sprintf(RspStr, "\r\n%s: SMS
PERIOD\r\n",AT_PERIOD_CMD);
                        ATRSP(RspStr);
                break;

                case ADL_CMD_TYPE_READ:

                        wm_sprintf(RspStr, "\r\n%s:%s
sn\r\n",AT_PERIOD_CMD+2,fPeriod);
                        ATRSP(RspStr);
                break;

                case ADL_CMD_TYPE_PARA:
                        param0=ADL_GET_PARAM(paras, 0);
                        if (wm_isnumstring(param0))
                        {
                                wm_memset((void*)fPeriod,0,sizeof(fPeriod));
                                wm_sprintf(fPeriod,"%.5d",wm_atoi(param0));
                                adl_flhWrite((ascii*)HND,iPeriod,
(u16)(wm_strlen(fPeriod)+1), (u8 *)fPeriod);
```

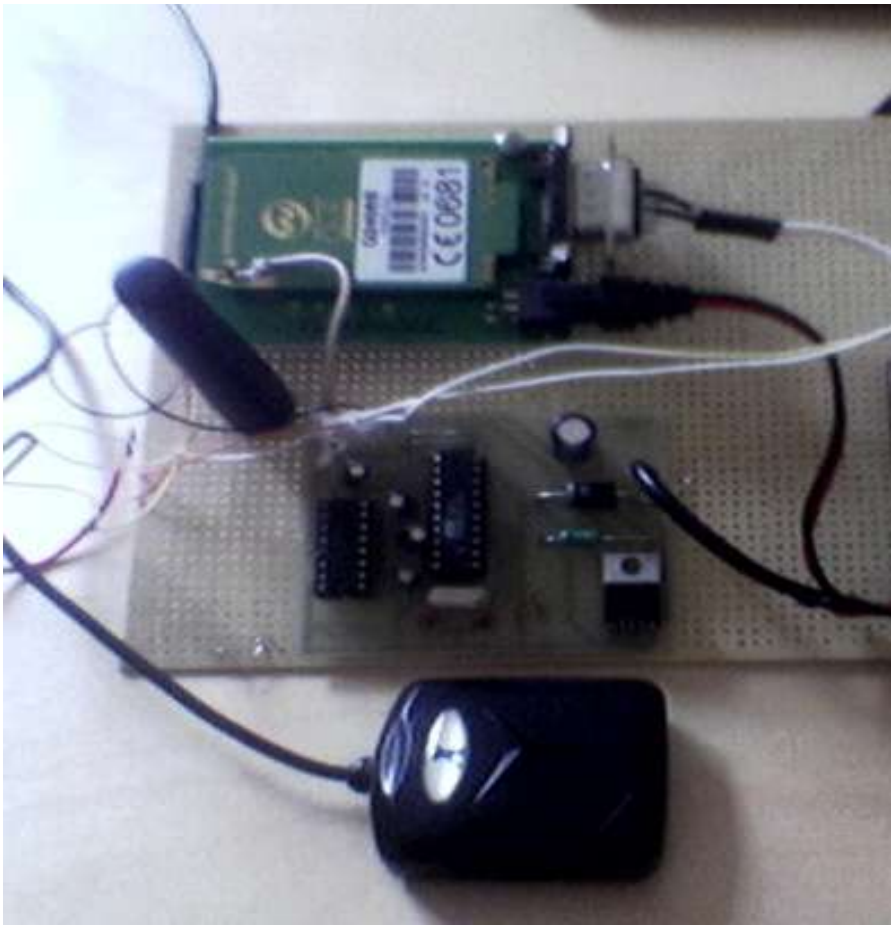
```
                    wm_sprintf(RspStr, "\r\n%s:%s
sn\r\n",AT_PERIOD_CMD+2,fPeriod);
                    ATRSP(RspStr);
                }
                else
                {

                    wm_sprintf(RspStr, "\r\n%s: belirtilen deger sn
cinsinden bir sayi olmali \r\n",AT_PERIOD_CMD+2);
                    ATRSP(RspStr);
                }
            break;
        }


                ATRSP("\r\nOK\r\n");

}

void AT_MERKEZ_CMD_Handler(adl_atCmdPreParser_t *paras)
{
    ascii *param0, RspStr[30];
    s32 iparam0;

    switch (paras->Type)
    {
        case ADL_CMD_TYPE_TEST:
            wm_sprintf(RspStr, "\r\n%s: SMS
MERKEZ\r\n",AT_MERKEZ_CMD);
            ATRSP(RspStr);
        break;

        case ADL_CMD_TYPE_READ:

            wm_sprintf(RspStr,
"\r\n%s:%s\r\n",AT_MERKEZ_CMD+2,fMERKEZ);
            ATRSP(RspStr);
        break;
```

```
                case ADL_CMD_TYPE_PARA:
                        param0=ADL_GET_PARAM(paras, 0);
                        wm_memset((void*)fMERKEZ,0,sizeof(fMERKEZ));
                        wm_sprintf(fMERKEZ,"%s",param0);
                        adl_flhWrite((ascii*)HND,iMERKEZ,
(u16)(wm_strlen(fMERKEZ)+1), (u8 *)fMERKEZ);
                        wm_sprintf(RspStr,
"\r\n%s:%s\r\n",AT_MERKEZ_CMD+2,fMERKEZ);
                        ATRSP(RspStr);
                break;
        }

                        ATRSP("\r\nOK\r\n");
}

void AT_GPS_CMD_Handler(adl_atCmdPreParser_t *paras)
{
        ascii *param0, RspStr[200];
        s32 iparam0;

        switch (paras->Type)
        {
                case ADL_CMD_TYPE_TEST:
                        wm_sprintf(RspStr, "\r\n%s: GPS\r\n",AT_GPS_CMD);
                        ATRSP(RspStr);
                break;

                case ADL_CMD_TYPE_READ:

                        wm_sprintf(RspStr, "\r\n%s:%s\r\n",AT_GPS_CMD+2,fGPS);
                        ATRSP(RspStr);
                break;

                case ADL_CMD_TYPE_PARA:
                        param0=ADL_GET_PARAM(paras, 0);
                        wm_memset((void*)fGPS,0,sizeof(fGPS));
```

```
                    wm_sprintf(fGPS,"%s",param0);
                    wm_sprintf(RspStr, "\r\n%s:%s\r\n",AT_GPS_CMD+2,fGPS);
                    ATRSP(RspStr);
                break;
        }


                    ATRSP("\r\nOK\r\n");

}

void At_Get_Params(void)
{

        u32 pLen;

        pLen=adl_flhExist((ascii*)HND,iPeriod);
        if (pLen>0)
        {
                adl_flhRead((ascii*)HND,iPeriod, pLen, (u8 *) fPeriod);
        }
        else
        {
                wm_sprintf(fPeriod,"00180");
                adl_flhWrite((ascii*)HND,iPeriod, (u16)(wm_strlen(fPeriod)+1), (u8
*)fPeriod);
        }

        pLen=adl_flhExist((ascii*)HND,iMERKEZ);
        if (pLen>0)
        {
                adl_flhRead((ascii*)HND,iMERKEZ, pLen, (u8 *) fMERKEZ);
        }
}
```

```
void adl_main ( adl_InitType_e  InitType )
{
	if (adl_flhSubscribe((ascii*)HND,10)==OK)
		At_Get_Params();
	else
		At_Get_Params();

	adl_atCmdSubscribe(AT_PERIOD_CMD, AT_PERIOD_CMD_Handler,
ATCMDTYPE);
	adl_atCmdSubscribe(AT_MERKEZ_CMD, AT_MERKEZ_CMD_Handler,
ATCMDTYPE);
	adl_atCmdSubscribe(AT_GPS_CMD, AT_GPS_CMD_Handler,
ATCMDTYPE);
	SmsHandle=adl_smsSubscribe((adl_smsHdlr_f )SMS_Handler,
(adl_smsCtrlHdlr_f) SMS_Control_Handler, ADL_SMS_MODE_TEXT);
	adl_tmrSubscribe(FALSE,30,ADL_TMR_TYPE_100MS,timerHandler1);
}
```





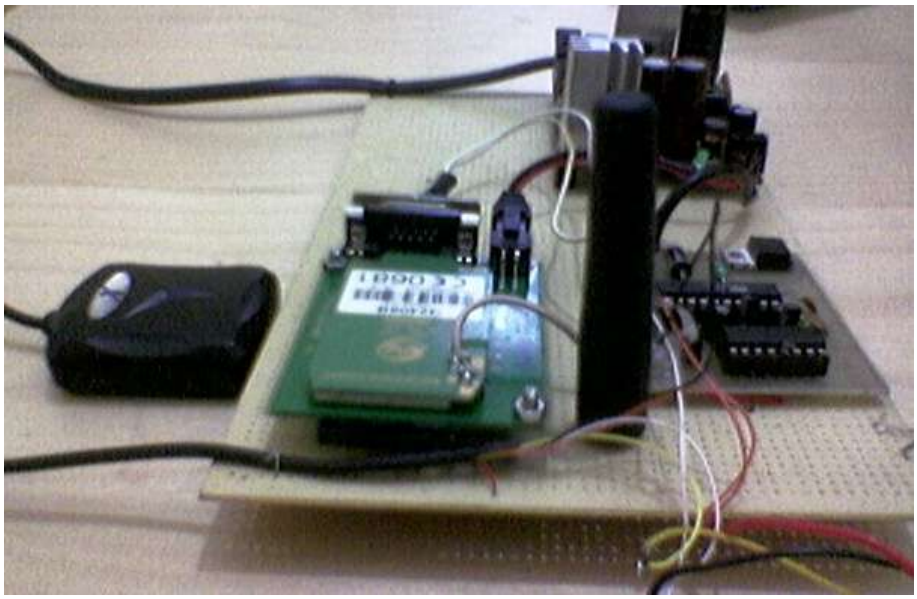


Figure 22 Device Picture

REFERENCES

ÖZCERİT, A.T. –ÇAKIROĞLU, M.- BAYILMIŞ,C (2005), **8052 Mikrodenetleyici Uygulamaları**,Papatya yayınevi, İstanbul.

SARIDOĞAN, Erhan (2007), **C++ ve Nesne Yönelik Programlama**,İstanbul

AYDIN, Tarkan (2004) **GSM ve GPS interfacing with Motorola Microcontroller** yayınlanmamış lisans tezi, Bahçeşehir Üniversitesi Bilgisayar Mühendisliği, İstanbul

ÇALIŞKAN, Enver (2004) **Online Vehicle Tracking by Using GPS and GPRS** yayınlanmamış yükseklisans tezi, Bahçeşehir Üniversitesi Fenbilimleri Enstitüsü, İstanbul

YOMRALIOĞLU, Tahsin (2001) **Cografi Bilgi Sistemleri ve Temel Kavramlar Uygulamalar Akademi yayınevi**,İstanbul

VIKIPEDIA Online Kütüphane,(2007)
http://en.wikipedia.org/wiki/Global_Positioning_System#Simplified_method_of_operation

GARMIN CORPARATION, (2007) , **http://www8.garmin.com/aboutGPS**

WAVECOM INC. (2002-2004), **P Fastrack Modem M1306B User Guide Datasheet.**

WAVECOM INC.(2002-2004) **P Wismo Quik Q2406 and Q2426 Product Specification Datasheet.**

KLEIN, Ramon (2003), Serial Library for C++
http://www.codeproject.com/system/serial.asp