ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

INVESTIGATION INTO, AND DEVELOPMENT OF, PERSONAL NAVIGATION SOFTWARE

Robin Morrison
BSc. (Hons.)

FOR REFERENCE ONLY

A thesis submitted in partial fulfilment of the requirements of The Nottingham Trent University for the degree of Master of Philosophy

April 1997

MOR Ref

ABSTRACT

This thesis describes and discusses an investigation into, and development of, a personal navigation software system. Current navigation technology and techniques are reviewed, from the traditional map and compass to the Global Positioning System and the electronic compass. This review highlights the deficiencies within current navigation technology and techniques, and establishes the possibility of creating a personal navigation device, part of which would be the software designed, developed and investigated in the thesis.

The design and development of the software is presented, illustrating that by translating those actions which the navigator currently carries out into an electronic equivalent, the Electronic Navigation Environment is created with four distinct modules: Map Display, Global Positioning System Driver, Tools and Software User Interface.

Systems testing assessed whether or not the Electronic Navigation Environment was operational and met the requirements of the User. The following tests were carried out:

- Map Display operation with vector and raster data.
- Global Positioning System Driver accuracy and static and mobile operation.
- Tool operation with vector and raster data.
- Software User Interface ease of use and ease of learning.

The results and analysis suggest that most aspects of the system are operational and meet the user requirements in terms of functionality, ease of use and ease of learning. However two main shortfalls are highlighted:

- The accuracy of the Global Positioning System Driver was +/- 102 metres which is not sufficient when using map scales larger than 1:50 000.
- The effective operation of the navigation software is reduced when colour raster data is displayed.

These shortfalls could be reduced or eradicated by adopting the following proposals:

- Module integration by upgrading the software used as a design platform
- Use of Differential Global Positioning Systems to improve driver accuracy

The research in this thesis illustrates that the Electronic Navigation Environment is a feasible and successful design which has the potential for future development and improvement.

OBJECTIVES

The main objectives of the research for this thesis were to:

(i) Carry out a literature review to investigate current navigation technology and techniques.

(ii) Establish a basis for the design and development of a Personal Navigation Device and Software.

(iii) Determine User Requirements of Personal Navigation Software.

(iv) Design and Develop Personal Navigation Software.

(v) Test Personal Navigation Software to assess whether or not it meets the requirements of the User in terms of functionality, operation and ease of use.

ACKNOWLEDGEMENTS

The author would like to thank the Department of Civil and Structural Engineering at The Nottingham Trent University for their support and use of resources throughout this research project.

Sincere thanks to my supervisors, Mr Philip Sargent, Senior Lecturer, Department of Civil and Structural Engineering, The Nottingham Trent University and Professor Max Robinson, Department of Electrical and Electronic Engineering, The Nottingham Trent University for their invaluable advice, guidance and support during the course of this research.

Thanks also to Miss Chris Wensley and Miss Debbie Manku for proof reading this thesis and for their constructive criticism.

Finally the author would like to thank the Estates Department at The Nottingham Trent University for supplying the digital map data of Clifton Campus required during testing.

LIST OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ASCII	American Standard Code for Information Interchange
CAD	Computer Aided Design
DOS	Disk Operating System
GP	Graphics Package
GIS	Geographical Information System
GPS	Global Positioning System
GUI	Graphical User Interface
NAVSTAR	Navigation Satellite Timing and Ranging
NMEA	National Marine Electronics Association
OSGB36	Ordnance Survey Great Britain 1936
OSGRS80	Ordnance Survey Global Reference System 1980
OSNG	Ordnance Survey National Grid
PDOP	Position Dilution of Precision
TAIP	Trimble ASCII Interface Protocol
TSIP	Trimble Standard Interface Protocol
SDK	Software Development Kit
SUI	Software User Interface
URT	User Response Time
WGS84	World Geodetic System 1984

CHAPTER 1: AN INTRODUCTION TO LAND NAVIGATION

1.1 What is Navigation?

Navigation is a fundamental process utilised by almost all animals. Many animals have a highly developed sense of navigation which allows them to undertake great migratory journeys (for example, the arctic tern and reindeer of Northern Alaska). In comparison to these animals, human beings have a poor sense of navigation and have resorted to the use of technology in order to overcome this deficiency.
Navigation can be defined as the art or science of determining position and orientation on, above or below the surface of the Earth, or, as defined by Keay (1989):

"... the highest level of navigation is the ability to find two places on a map, determine a course which will lead from one to the other and then be able to make that journey."

This definition of navigation explains the process as used by most land navigators. There are simpler approaches to navigation, such as the visual approach (i.e. the navigator can see his or her goal and works their way towards it) or pilotage, which relies on the navigator having a sound knowledge of the surrounding area and travelling from one known point to the next in order to reach his or her goal (i.e. the map is memory based as opposed to paper based). These simpler approaches are by their nature of limited use, requiring knowledge of the area or visibility of the target destination. The definition quoted above is the most applicable to the land navigator, although the simpler approaches can also be applied to the varying techniques which the land navigator employs. Regardless of how it is defined, Hartley (1996) identified three fundamental elements constituting navigation:

- Where is a particular location, place, oneself etc.?
- What is at a specified location, co-ordinate etc.?
- How do I get there?

1.2 Navigation Tools and Technology

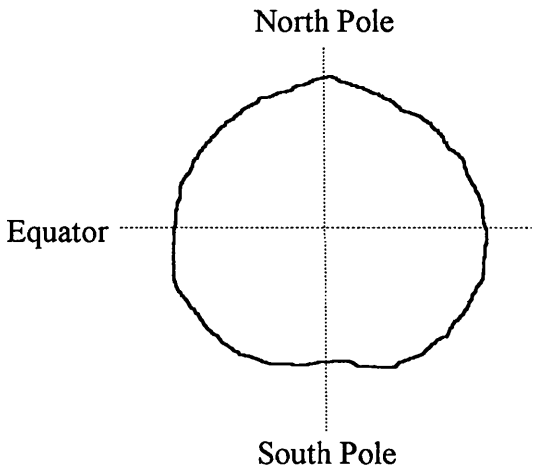
This section discusses the tools and technology used by man to aid and enhance his navigational abilities on land, from the traditional tools of map and compass to the use of satellite technology.

1.2.1 Map

The paper map is the most vital piece of equipment that navigators have access to at present. It allows them to perform many tasks including determining position in relation to their surroundings and also route planning.

A map is a two-dimensional graphical representation of the Earth's surface. Every map is based on one of the many projection systems available which allow a three-dimensional spheroidal surface to be represented on a two-dimensional piece of paper. In addition, a co-ordinate system provides a reference frame which allows the map user to define any point on the map numerically and unambiguously. Maps used to represent Great Britain are based on the Transverse Mercator Projection and a co-ordinate system known as the Ordnance Survey National Grid (OSNG).
It is important to understand the relationship between the Earth's surface, the projection system and co-ordinate system.

The Earth is a planetary body that revolves round the Sun. As stated by Bugayevskiy and Snyder (1995) the shape of the Earth was thought to be spherical up until the eighteenth century, when it was suggested that the Earth was an oblate spheroid due to its rotation and the gravitational attraction of the Sun and Moon (Leick, 1990) (Figure 1.1). More recently, data has been collected on the effects of gravity on artificial satellite orbits to suggest that the Earth is 'pear-shaped', with the southern hemisphere slightly larger than the northern hemisphere (Vanicek and Krakowski, 1986).



'Pear-Shaped' Earth derived from the effects of gravity on satellite orbit

Figure 1.1 Shape of the Earth

The surface of the Earth is irregular and uneven and known as the topographic or physical surface (Cross *et al*, 1989). The topographic surface includes continental bodies and ocean floors and varies in height by up to 10km between its highest and lowest points. This random nature of the Earth's surface means that it cannot be mathematically defined.

The Earth has another physical surface known as the geoid. As discussed by Methley (1991) and Seeber (1993) the geoid is the equipotential surface of the Earth, which coincides on average with mean sea level. The geoid does not coincide exactly due to the effects of wind, tides and currents on sea level. Although the geoid is a much smoother surface than the topographic surface it still cannot be mathematically defined. The nature of these two physical surfaces of the Earth means that in order to determine position it is necessary to use a mathematical model of the Earth's surface. This model is known as the reference ellipsoid (Figure 1.2).

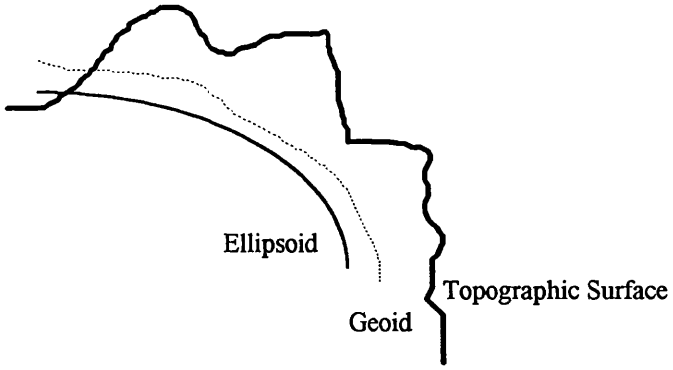
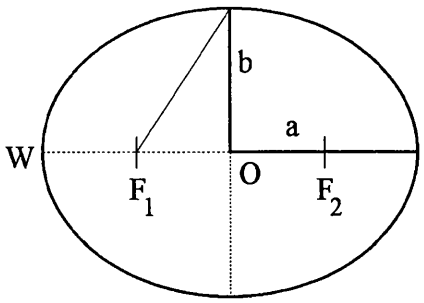


Figure 1.2 Earth's Surfaces

The reference ellipsoid defines a frame of reference to which positions determined on the Earth's surface can be related. The ellipsoid allows the use of a co-ordinate system as any point on the ellipsoid can be defined exactly. As stated by Ewing and Mitchell (1970) an ellipsoid can be described as an ellipse that is rotated about its minor axis, with its major axis forming the equatorial plane. The ellipsoid can be mathematically defined in terms of a semi-major and semi-minor axis (a and b), flattening (f) and eccentricity (e) (Figure 1.3).

Ellipsoids have been used to define the Earth on a global, regional and local level. This is because certain ellipsoids coincide more closely with the geoid in a particular region than other ellipsoids, which may be better suited to a different region or to the Earth as a whole.



Where a = semi major axis

b = semi minor axis

F_1, F_2 = Foci

$f = \text{flattening} = \frac{(a-b)}{a}$

$e = \text{eccentricity} = \frac{OF_1}{OW}$

$e^2 = \frac{a^2 - b^2}{a^2}$

Figure 1.3 Ellipsoid Definition
Source: Ewing and Mitchell (1970)

The vast majority of mapping in Great Britain is based on a local ellipsoid known as Airy's Spheroid (N.B. The terms ellipsoid and spheroid are interchangeable). Airy's Spheroid can be mathematically defined by:

a = 6377563.396
b = 6356256.910
$f = 3.340850522 * 10^{-3}$

The models used to define position on the Earth so far in this thesis are three-dimensional. However, a map is a two-dimensional surface and in order to relate the three-dimensional position defined on the ellipsoid to the two-dimensional map a projection system is used. The projection system used in Great Britain on most maps is a modified form of the Transverse Mercator.

According to Raisz (1948) and the Ordnance Survey (1983) the simplest way to understand this projection is to relate it to a horizontal cylinder that touches the ellipsoid used to represent the Earth along a meridian (a meridian being any plane that passes through both poles of the ellipsoid along its surface). Points can then be transferred from the ellipsoid to the cylinder (this is carried out mathematically). The cylinder can then be unrolled to form a flat surface (Figure 1.4).

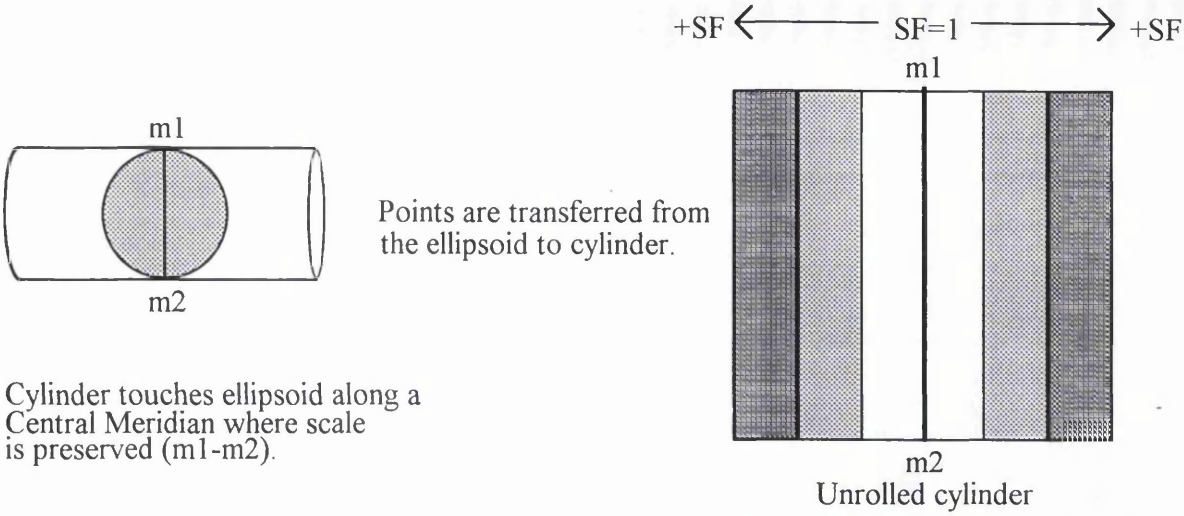


Figure 1.4 Transverse Mercator Projection

When transferring points from the three-dimensional surface of the ellipsoid to the two-dimensional map surface, distortions are incurred as the surfaces are nonapplicable (Robinson *et al*, 1985). In the case of the Transverse Mercator projection, scale is preserved along the central meridian (where the scale factor is equal to 1) but increases rapidly away from it, making the projection suitable for mapping areas with greater extents in latitude than longitude (Ordnance Survey, 1950). According to Maling (1992) if the Transverse Mercator were used in Great Britain, its east-west extents (from 1 degrees 43 minutes East at Great Yarmouth to 8 degrees 34 minutes West at St. Kilda) are great enough to create significant distortions in scale, as the scale factor would increase around St. Kilda to 1.0018.

In order to minimise these distortions, the Transverse Mercator used in Great Britain is modified. The Ordnance Survey (1983) state that the scale factor along the central meridian is reduced to 0.99960127 which has the effect of creating two lines of zero distortion (where the scale factor is 1) (Figure 1.5). Distortions at the east-west extents of the projection system are reduced to a scale factor of approximately 1.0004. Therefore, the reduction of the scale factor along the central meridian extends the zones of mapping without increasing distortion, compared with employing a central meridian with a scale factor of 1.

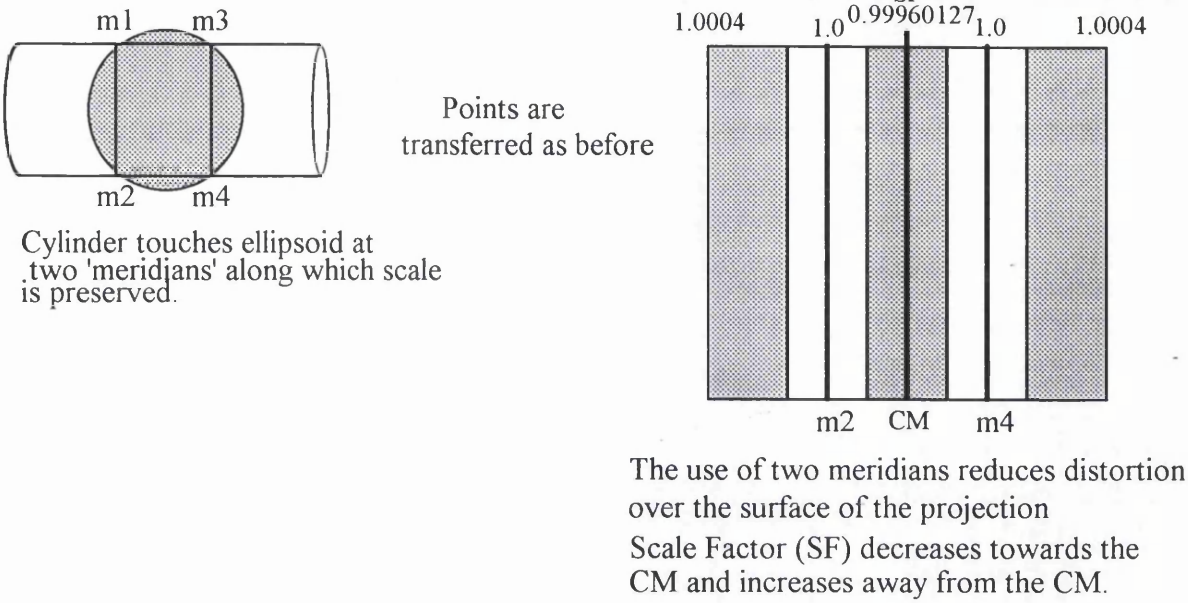


Figure 1.5 Modified Transverse Mercator used in Great Britain

A plane rectangular co-ordinate system is used to define position on most maps of Great Britain. The system uses two axes to define position in terms of an Easting and a Northing, thus forming a grid covering Great Britain. This grid is known as the Ordnance Survey National Grid (OSNG). The true origin of the system is located at 49 degrees North of the equator and 2 degrees West of the Greenwich Meridian on Airy's spheroid (Figure 1.6). Using the true origin would provide negative co-ordinate values for areas of Great Britain west of the Greenwich Meridian (Harley, 1975). To solve this problem a false origin is used so that all co-ordinate values defined within the grid system are positive. This system is easy to define and to visualise making it suitable for everyday use.

The OSNG co-ordinate system is used by the majority of maps in Great Britain, including those used by mountaineers, e.g. the Ordnance Survey Landranger Series, Bartholomew's, and Harvey's walking maps.

Paper maps for land navigation come in many forms and scales and use many different conventions. They do, however, display the same basic information (Keates 1989):

- Point Symbols
- Line Symbols
- Area Symbols
- Relief

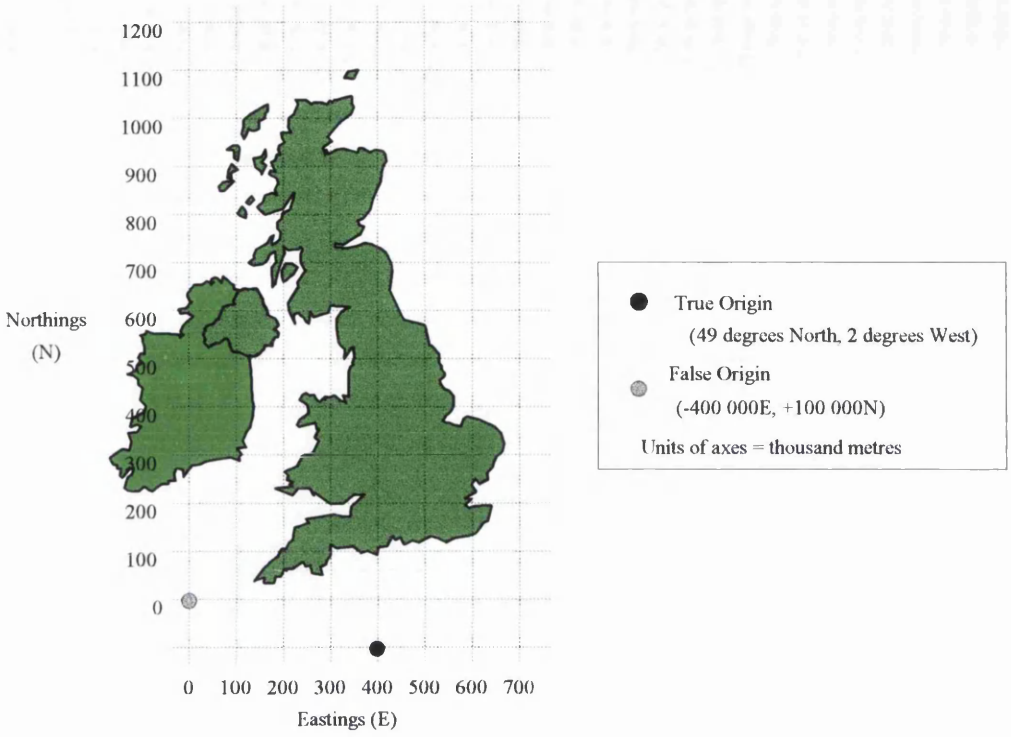


Figure 1.6 Ordnance Survey National Grid
Source: Harley (1975)

Point Symbols represent features that occur in a single location on the Earth's surface. For example, this may be a building, telephone box, or triangulation pillar. Point symbols can also be used to represent relief in the form of spot heights (Figure 1.7).

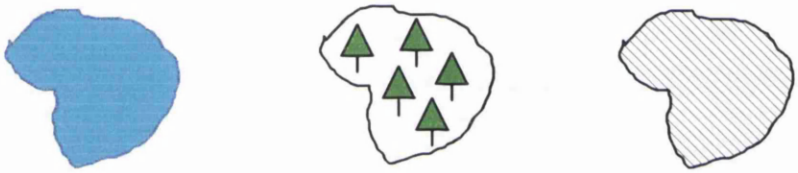
Figure 1.7 Point Symbols

Line Symbols are used to represent linear features. There are two main types of linear features: those that represent a physical feature (e.g. a river or road) and those that represent non-physical features (e.g. county boundaries, national park areas). These two types of line symbol can often coincide (e.g. where a river is used to denote a county boundary). See Figure 1.8 for examples of line symbols.

River | County Boundary | County Boundary coinciding with river.

Figure 1.8 Line Symbols

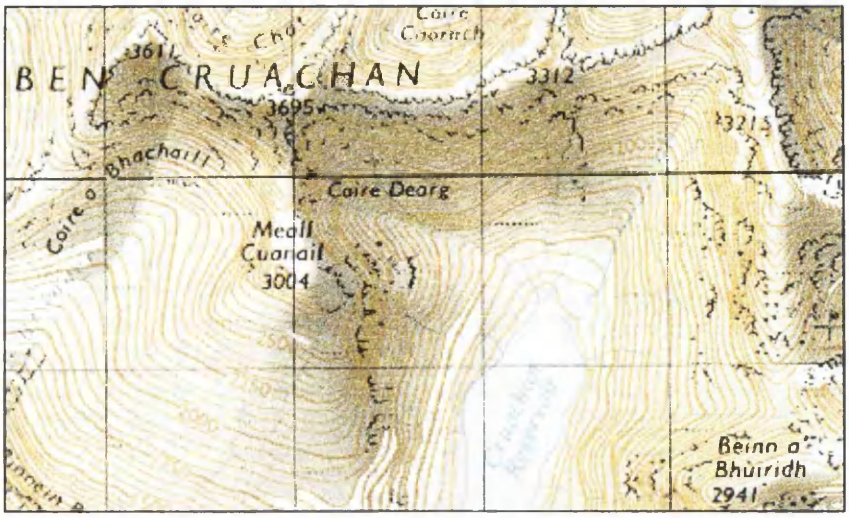
Area symbols represent large physical features such as woodland, lakes, or urban areas. They are usually infilled with solid colours, tints or patterns (Figure 1.9).



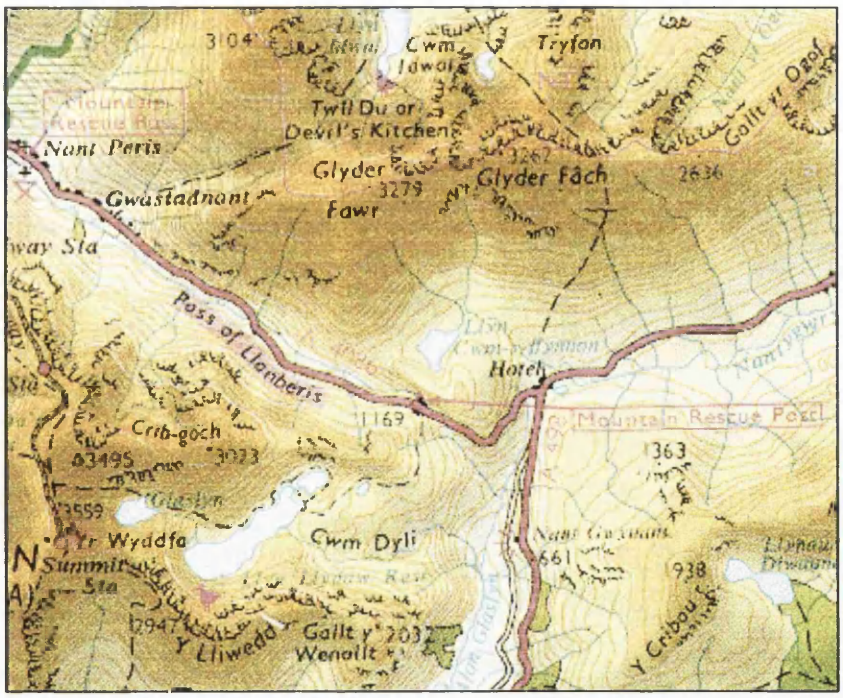
Lake | Woodland | Orchard

Figure 1.9 Area Symbols

One of the most important features represented on maps (from the hillwalker's point of view) is relief, as often, in remote regions, point, line and area features are scarce. Relief can be represented in many ways, one of the most common being contour lines. These are essentially isolines connecting points of the same height, which build up to provide a two-dimensional representation of relief. Other methods that can be used are hill-shading, hypsometric colours (bands of colours representing fixed height intervals), hachures (representing relief by defining slope, little used now except for specialist features), and spot heights. Each method has its own advantages and drawbacks, and because of this are often used in various combinations (Robinson *et al*, 1985). Figure 1.10a shows the use of contours and hill shading. Figure 1.10b shows the use of hypsometric colours, contours and spot heights.



a) Contours and Hillshading



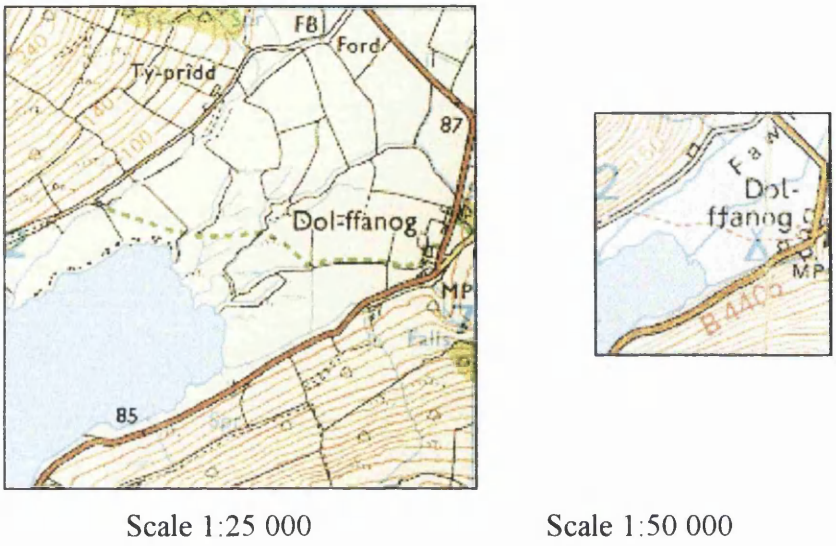
b) Contours and Hypsometric Colours

Figure 1.10 Relief Representation

© Crown copyright

As discussed by Wale (1995) the level of detail displayed on maps varies with scale and manufacturer. Some manufacturers provide maps custom-made for navigating in wilderness environments (e.g. Harvey's) which provide very detailed representations of physical features. Other maps, such as the Ordnance Survey Landranger, are designed more for general purpose use and therefore provide less detailed representations of wilderness areas.

Scale also affects the level of detail shown on maps. Smaller scale maps can show greater area but in less detail than larger scale maps, which cover smaller areas of the Earth's surface (Figure 1.11).



Scale 1:25 000 Scale 1:50 000

Figure 1.11 Map Scales

Figure 1.11 shows two maps covering the same area at different scales, with the difference in the degree of detail (note especially field boundaries) evident.

1.2.2 Compass

The compass is designed to provide a means of determining orientation without reference to the surrounding terrain. The operation of the compass is based on the magnetic field that surrounds the Earth. This field was discovered and used to develop the magnetic compass in the 12th century AD by the Chinese (Walker S, 1995).

As explained by Press & Siever (1985), the magnetic field of the Earth runs from the Magnetic North Pole to Magnetic South Pole. The Earth's magnetic axis is inclined from its geographic axis by approximately 11 degrees. If a piece of magnetic material is placed on a pivotal point, it will align itself with the Earth's magnetic field, i.e. north-south. If this principle is applied to navigation, it can provide the user with a north direction at any point on the Earth (Figure 1.12).

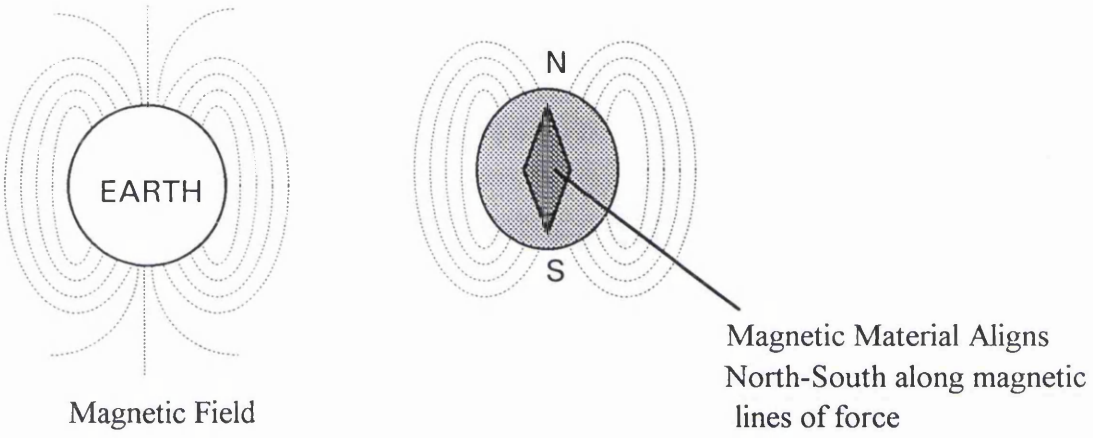
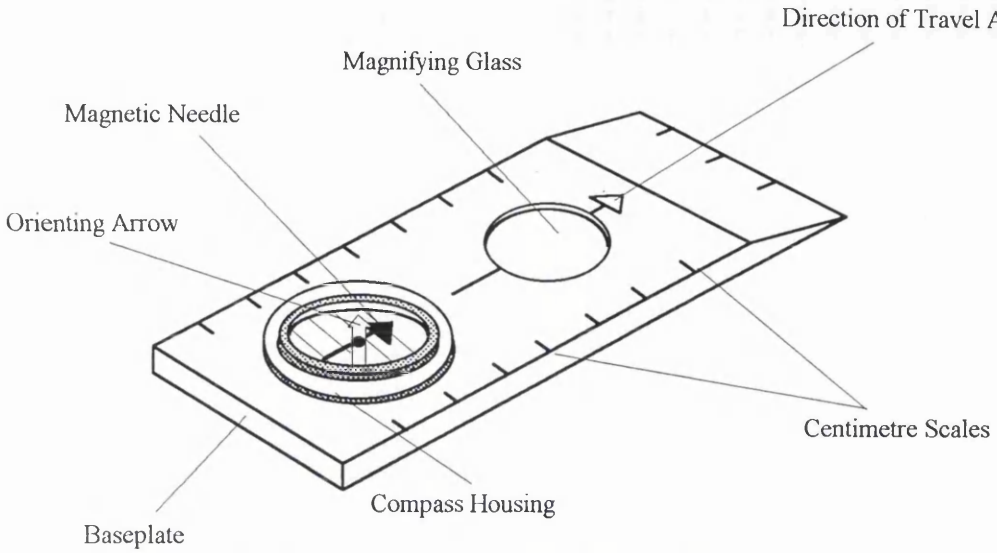


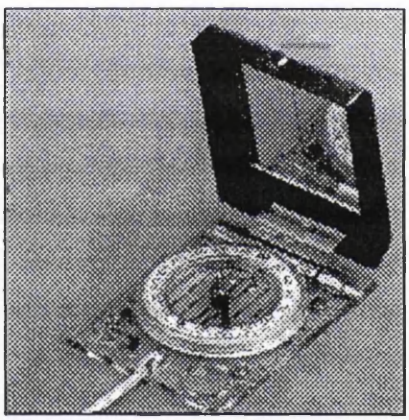
Figure 1.12 Compass

The modern compass has since been developed, and usually incorporates a sexagesimal reading system (Bannister *et al*, 1992) used to define the navigator's orientation as a bearing. Some examples of the modern compass are shown in Figure 1.13.



a) Diagramatic view of a typical navigator's compass

Source: Keay (1989)



b) Silva Model MC5 Mirror Sighting Compass

Figure 1.13 Magnetic Compass

A recent development has been the electronic compass. Originally utilised in maritime and aeronautical fields, the technology has been developed into a hand-held unit for the land navigator. An electronic compass can operate in the same fashion as the traditional compass, or can give a digital readout of bearing, as well as storing route and waypoint information (Figure 1.14).

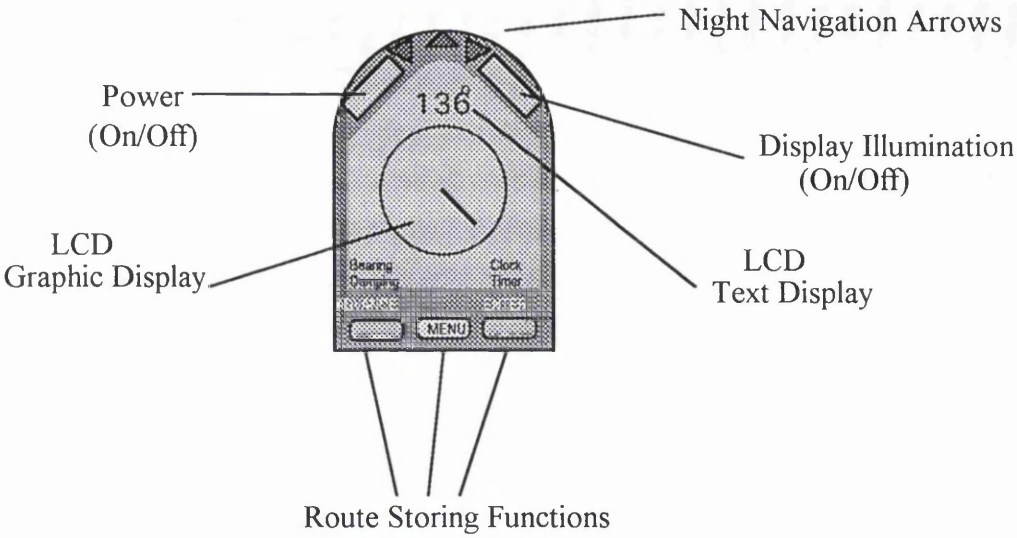


Figure 1.14 The Wayfinder Electronic Compass

1.2.3 Chronometer

As stated by Keay (1989) a chronometer or time piece is another important piece of navigational equipment. It allows the navigator to calculate speed over ground and to determine approximate times of arrival as well as distances. Typically a wristwatch is suitable for navigation, although a stopwatch or more accurate timepiece can be used. The accuracy of the time piece is dependant on many factors including the task the timepiece is being utilised for, the nature of the terrain, and personal preference.

1.2.4 Light Source

A light source is required when navigating in poor visibility or at night. This is necessary for reading the map and in some cases reading the compass (if no luminous dial is present). According to Keay (1989) it is important that the light source is not so bright as to destroy the navigator's night-vision. Nevertheless, it must produce sufficient light for the navigator to read maps. Typical light sources must be fully portable and will usually be a battery-powered hand-held torch or headlamp.

1.2.5 Global Positioning System

The Global Positioning System (GPS) is the most recent piece of technology to be utilised by the land navigator. GPS (also known as NAVSTAR - NAVigation Satellite Timing And Ranging) is a military satellite positioning system originally designed for multipurpose navigation (Kennie and Petrie, 1993). It was then developed for civilian use, most notably by land surveyors. The rapid development and miniaturisation of the technology has seen an explosion in its use in almost every field where position is of any importance, from vehicle tracking, to monitoring the movements of the Earth's crust, to personal land navigation.

The GPS system consists of a constellation of 24 satellites which orbit the Earth. The satellites transmit ephemeris information, which defines their position in space at a specific time. A receiver on the Earth picking up these signals from three or more satellites can determine its position on the Earth's surface, in the same frame of reference as the satellites themselves. King *et al* (1987) states that GPS position can be determined using two methods: Carrier Phase Measurement and Pseudo Ranging .

Carrier Phase Measurement operates by determining the number of wavelengths in the signal between satellite and receiver, while pseudo ranging as discussed by Ackroyd and Lorimer (1990) is achieved by calculating the time taken for the ephemeris signal to travel from satellite to receiver. Since the wavelength of the signal is known, the distance can be calculated. Once the distance to three or more satellites has been calculated, then position can be determined by trilateration (Figure 1.15). The method predominantly used for navigation is pseudo ranging.

A single receiver can achieve a positional accuracy of +/- 25 metres but military signal downgrading (called Selective Availability) means the actual achievable accuracy is +/- 100 metres (Gilbert, 1996a). This +/- 100 metres accuracy can be improved using differential GPS (DGPS), where corrections for the downgrading can be broadcast on an FM carrier wave or sent from another receiver via a cellular phone network.

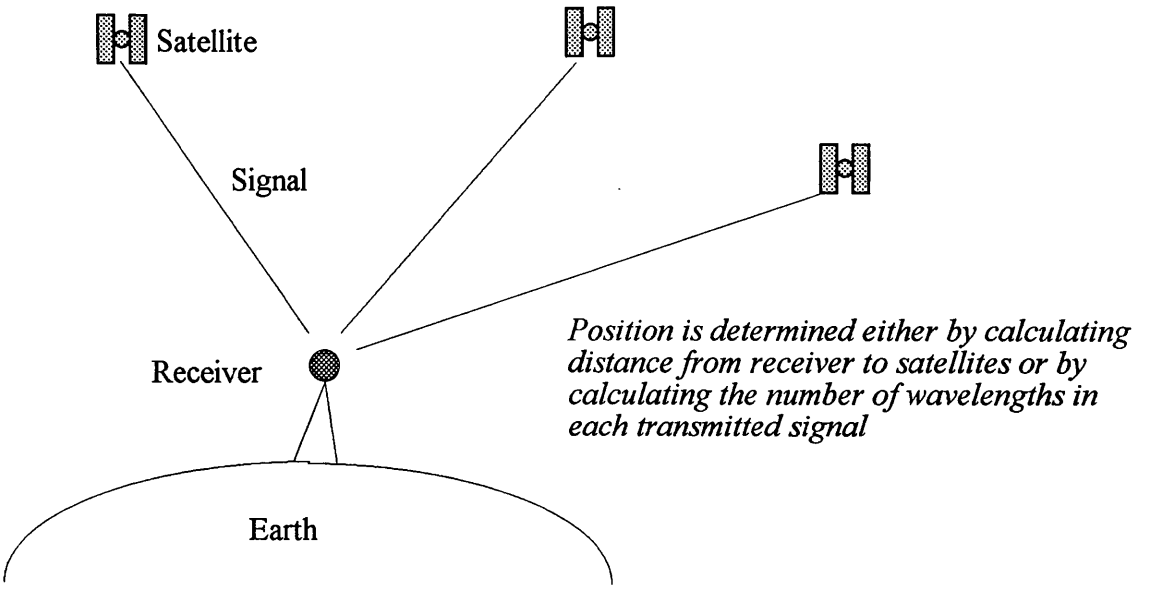


Figure 1.15 Global Positioning System

The accuracy achieved by a GPS receiver will also depend on satellite geometry. This was described by Gilbert (1996b) as "The physical configuration in the sky of the satellites used [by the receiver]". If the satellites are spread across the sky then the angle of intersection of the signals can be defined more precisely than if the satellites are grouped closely together (Figure 1.16).

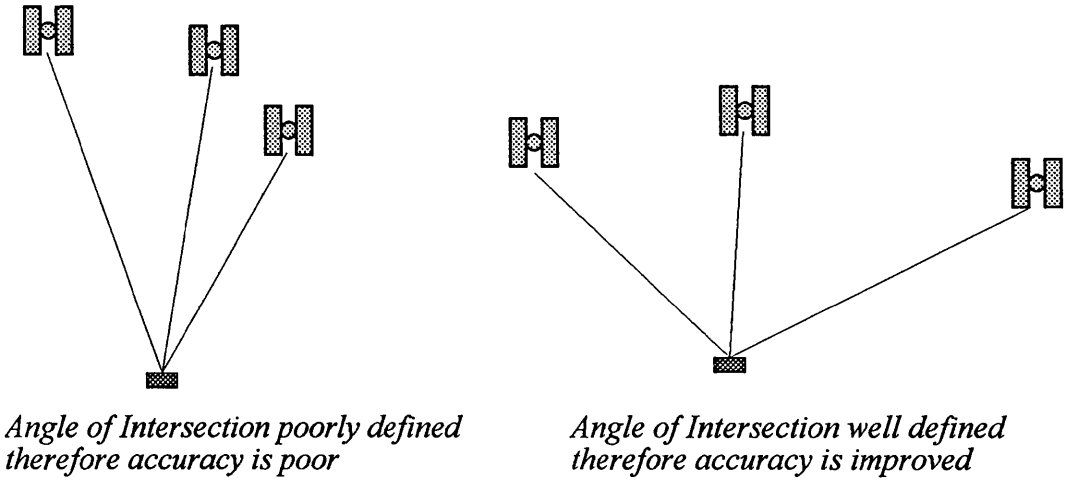


Figure 1.16 Satellite Geometry
Source: Gilbert (1996)

The ellipsoid used to define co-ordinates obtained through satellite observations is WGS84 (World Geodetic System 1984). This ellipsoid must be able to define any position on the Earth and so parameters have been used to provide a global ellipsoid:

a = 6378137
b = 6356752.3141
f = $3.352810703 * 10^{-3}$

The origin of WGS84 is the Earth's centre of mass, and position can be defined on the ellipsoid using Geodetic co-ordinates and Cartesian co-ordinates. Cartesian co-ordinates employ XYZ planes to define position (Dodson, 1995). The XYZ planes are mutually perpendicular and form three mutually perpendicular axes as they intersect in pairs.
As discussed by Kumar (1993) WGS84 Cartesian co-ordinates can be defined as follows. The Z-axis coincides with the Conventional Terrestrial Pole (CTP), as defined by Bureau International de L'Heure (BIH) 1984. The X axis defines a plane through the zero meridian as defined by BIH. Finally, the Y axis is at 90 degrees to both the Z axis and the X axis (Figure 1.17).

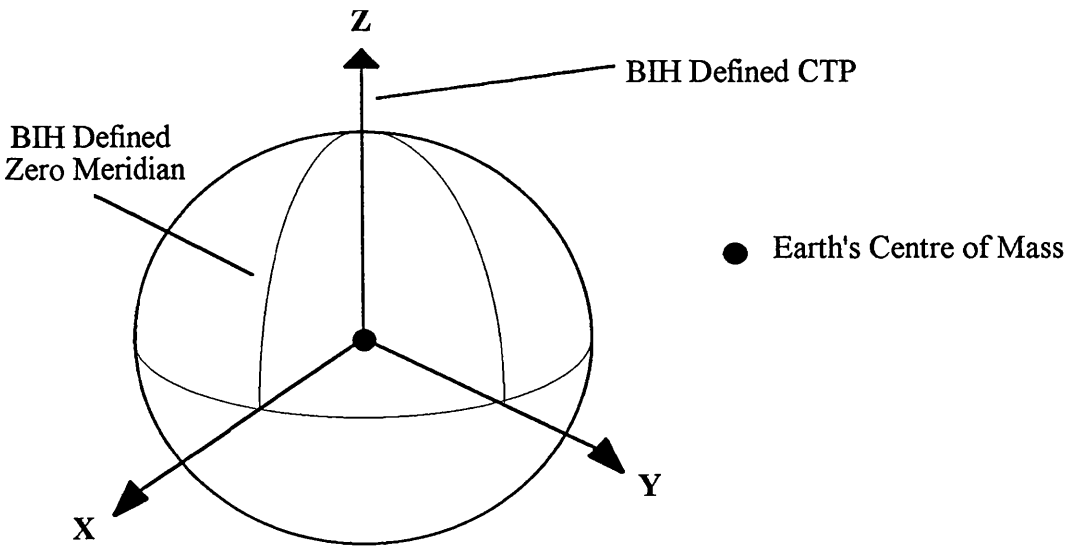


Figure 1.17 WGS84 Cartesian Co-ordinate Reference Frame
Source: Defense Mapping Agency (1991)

Cartesian co-ordinates can define any position on WGS84 in three-dimensions. However, to the typical map reader, cartesian co-ordinates are difficult to visualise in terms of their surroundings. Relating co-ordinate information to one's position on the Earth has proven easier using Geodetic co-ordinates.

Geodetic co-ordinates define position on WGS84 in terms of Latitude and Longitude. Latitude can be defined as the angle between the plane through the position point (P) and the plane of the equator (Figure 1.18). Longitude is the angle defined between two meridian planes, one passing through the positional point and the other being the zero meridian plane as defined by BIH. Most map users and navigators are familiar with Geodetic co-ordinates and they can more easily visualise their position using this system. This is due to the widespread use of Geodetic co-ordinates in navigation and mapping.

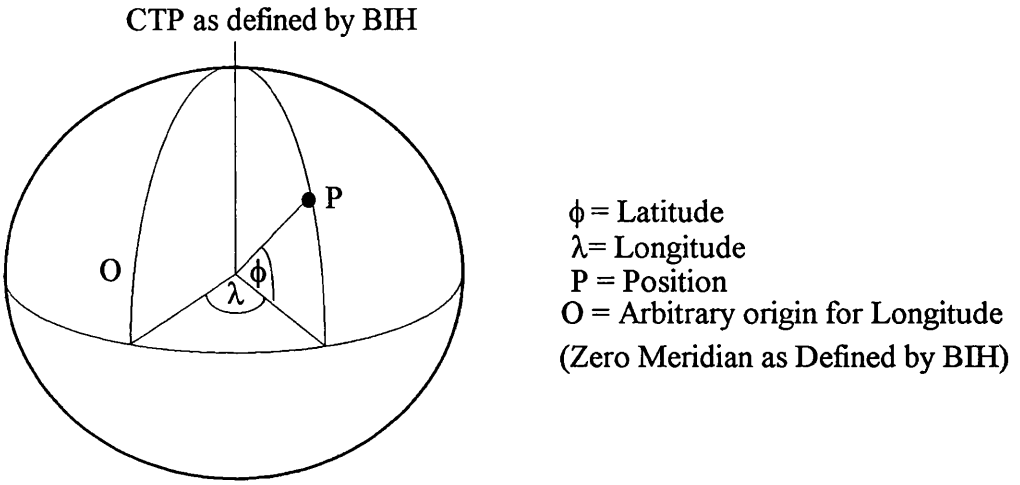


Figure 1.18 WGS84 Geodetic Co-ordinate Reference Frame

Handheld GPS units have been available for some time, for use mainly by the military as land navigation tools (Starbuck, 1995). In recent years the devices have found many uses including surveying and mapping, utility location, and, as illustrated by Archdeacon (1995) and Wright (1995) land navigation for recreation. These devices are pocket sized and will determine the user's position to an accuracy of one hundred metres when Selective Availability is turned on (Figure 1.19).

Essentially these devices provide a set of co-ordinate values which must then be related to a paper map so that the user can establish his position in relation to his surroundings.

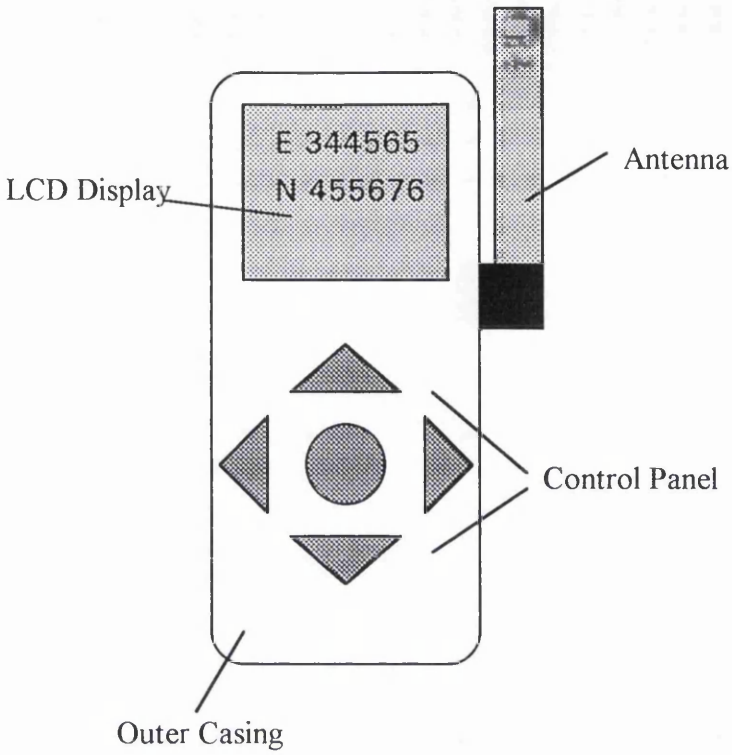
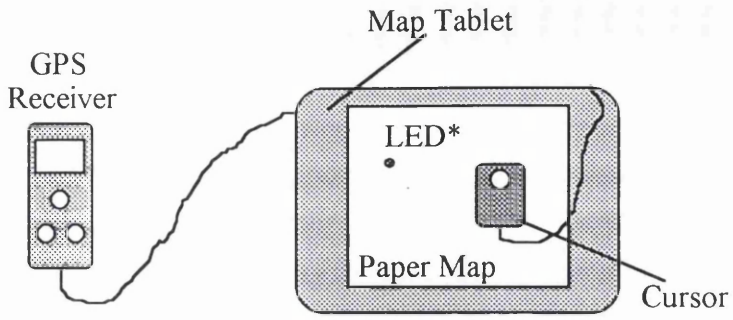


Figure 1.19 Hand-Held GPS

A number of organisations have attempted to integrate GPS and paper map by developing a map cursor which can be connected to the GPS receiver (e.g. the Silva Navimap Yeoman and the Azimuth Pointer Mk II). When a position is obtained, the cursor is driven or moved manually to the relevant position on the paper map, which is highlighted by a flashing LED on the map tablet (Figure 1.20).



**Flashing LED indicates current position as determined by GPS reciever. Cursor can be used to plot routes and waypoints.*

Figure 1.20 Hand-Held Receiver and Map Cursor

The GPS system has an added feature, which is that the satellies transmit their signals with reference to a very closely controlled time-frame. This enables the receiver to display current time, as well as calculate speed over ground and estimated time of arrival.

1.3 Navigation Methods

Now that the term 'navigation' has been defined, and the tools of navigation introduced, the methods of navigation can be explained. There are a multitude of methods available to aid in the process of navigation, using various combinations of the tools described in the previous section.

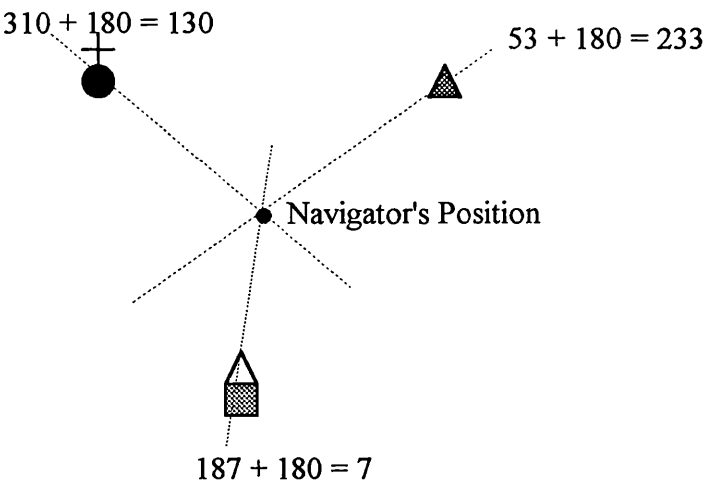
One aspect of land navigation is the art of determining position and orientation in unknown terrain. The position determination is carried out either in relation to surrounding features and landmarks, or to a co-ordinate system which is then related to the physical surroundings. Orientation is established in relation to surroundings or a bearing which allows orientation in relation to magnetic north. Land navigation differs from marine and air navigation in that the surface on which navigation takes place is stationary and the user either stationary or very slow moving, and so complex equations and mathematics are not required in order to determine position. One of the most fundamental ways of navigating on land is through map and compass.

1.3.1 Map and Compass

The map and compass can each be used independently in order to navigate but they are much more effective when used together.

Determining position can be carried out using the map alone or map and compass. As stated by Walker (1986) the essential skill required in order to be able to determine position using a map alone is map interpretation, i.e. the ability to relate what is on the map to the physical surroundings. One map interpretation skill which is extremely useful to the land navigator is terrain visualisation. This involves the ability to model the terrain in the mind from the relief representation on the map, and then relate that to the surrounding terrain form. This technique is important to the land navigator as often there are few suitable point or line features available which are easy to relate to.

Position can be determined in a number of ways using a map and compass. Point features are ideal for determining position as they can be clearly sighted using a compass. Using the resection method described by Keay (1989) three or more points should be sighted and, by calculating the back-bearing from the points (i.e. sighted bearing + 180 degrees) and plotting them on the map, the point of resection will give the navigator's position (Figure 1.21).



At least 3 points are sighted with a compass. Adding 180 degrees to the bearing gives the back bearing of each sighting. Resecting the 3 back bearings on a map gives the navigator's position

Figure 1.21 Resection by Back-bearing

Position can also be determined if the bearing to the last known point is known and the approximate distance travelled can be calculated. The accuracy of this method depends on the ability of the navigator to calculate their distance travelled with precision.

Walker (1986) states that route planning is another vital skill of the navigator and relies heavily on map interpretation in order to choose a suitable course. This includes the ability

of the navigator to use point features as waypoints (points at which course changes direction) and their ability to follow linear features (or use them as 'handrails' as described by Walker (1986)).
Once the navigator can master these skills, they can plot a suitable route on the map with bearings and approximate distances to follow, which if used correctly will greatly reduce the chance of the navigator becoming lost or disorientated.

Orientation can be carried out using a map, compass or both. Orientation can be determined using a compass in relation to magnetic north. It allows the navigator to follow a straight line route or determine the direction in which they are facing. With the aid of a map, orientation can be determined in relation to physical features represented on the map by transferring the bearing onto the map grid-system.
Orientation can also be determined using the map only, and, according to Keay (1989) this is best achieved using linear features, such as a river or fence. It relies on the navigator being able to identify where along the feature he or she is and then rotating the map until it matches the orientation of the physical feature (Figure 1.22).

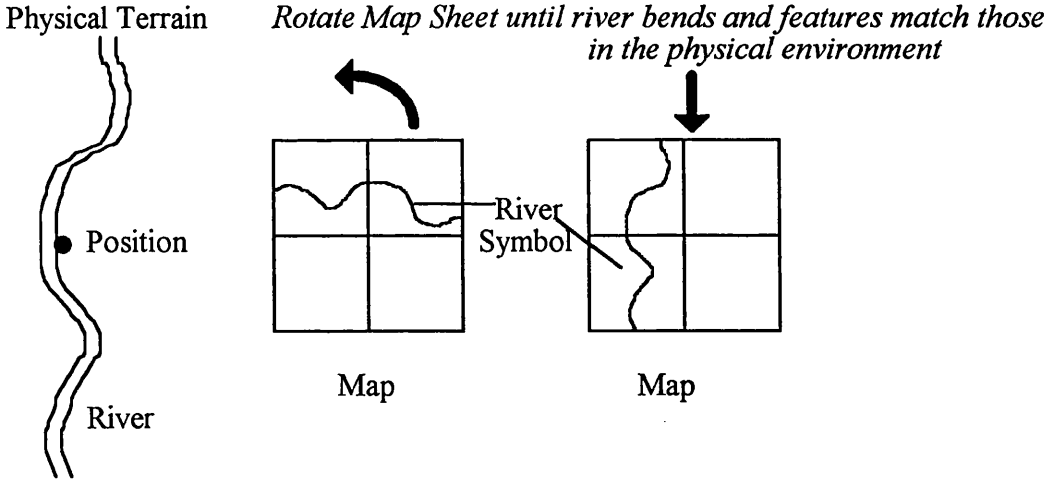


Figure 1.22 Orientation using Linear Map Features

1.3.2 Map and GPS

Archdeacon (1995) highlights that with the introduction of GPS, position can be plotted directly onto the map from the GPS receiver - as long as the same co-ordinate system is being used by both map and GPS. Many receivers have built in datums to allow transformation from WGS84 to local co-ordinate systems.

1.3.3 Stars and Sun

Navigation by the stars and sun has been practised for thousands of years and the fundamental principles remain unchanged. The stars and sun are most commonly used to find direction (or orientation) in land navigation.

Orientation can be determined using the sun and a watch. According to the methods outlined by Walker (1986) and Keay (1989), in the UK, the watch must be set to Greenwich Mean Time and placed on a flat surface. The watch is then rotated until the hour hand points directly towards the sun. A line bisecting the angle between the hour hand and twelve o'clock is positioning approximately south (Figure 1.23). This method is only approximate due to the complex movement of the sun and sighting along the hour hand of a watch. This can create errors of up to 25 degrees, therefore the accuracy of the time on the watch is not crucial for this method.

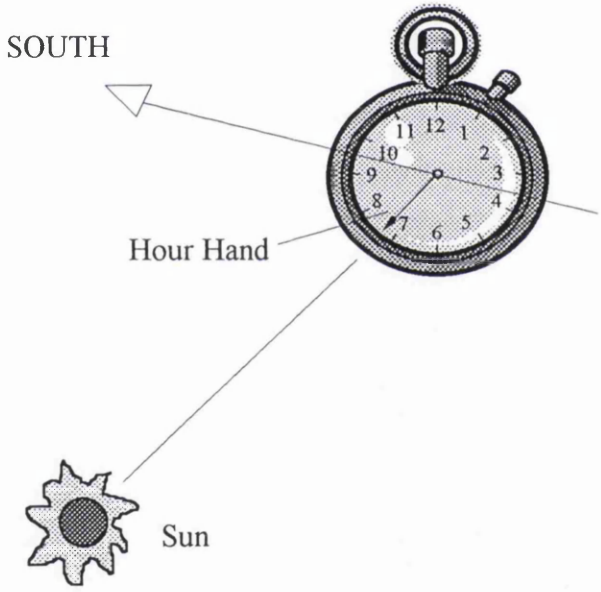


Figure 1.23 Determining Orientation Using the Sun

Orientation can also be determined by using the stars. The Pole Star (Polaris) is that most commonly used by navigators as it is located over the North Pole and only deviates from true north by +/- 2 degrees. This method provides a quick and easy way of determining North.

The stars can also be used with a compass to allow the navigator to follow a course or bearing. Selecting a star that lies on the direction of travel, a bearing to that star can be established using a compass. As stated by Keay (1989) the stars appear to rotate in relation to the Earth and will move approximately 15 degrees every hour. This means an approximate change in direction to the star of 5 degrees every 20 minutes and so either the

direction of travel can be altered accordingly or a new star along the original direction of travel can be selected (Figure 1.24).

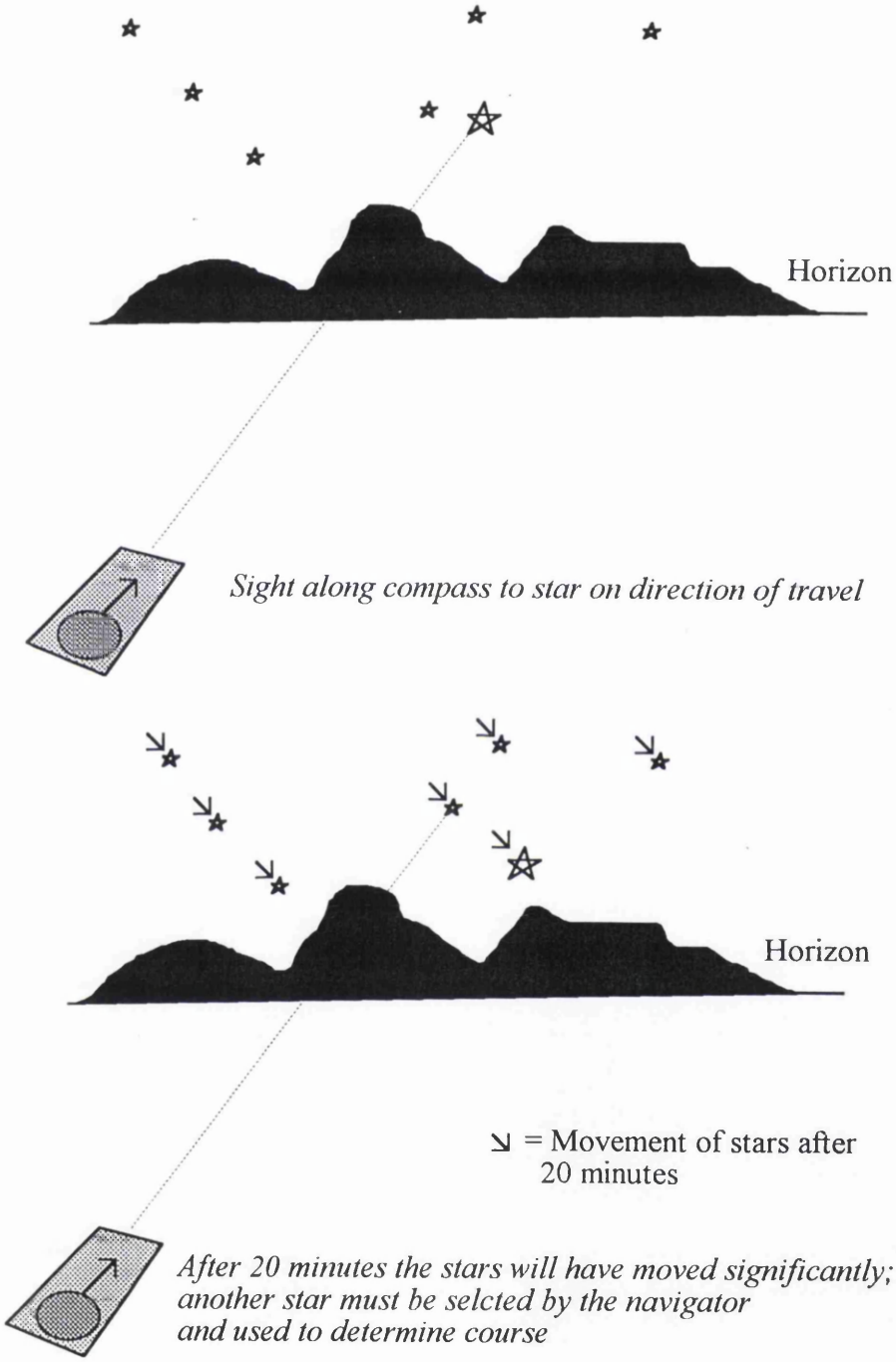


Figure 1.24 Determining Orientation Using Stars

The sun and stars are best used to orientate the navigator in conjunction with a map and compass when physical features prove unsuitable for such purposes. Walker (1989) states that stars provide ideal points of reference when navigating at night and the surrounding landscape cannot be seen.

1.4 Conclusions

Navigation can simply be defined as the determination of position and orientation and the ability to use this information in order to get from one point to another over unknown terrain. Whether or not man possesses an inbuilt navigation system as many animals do (a so-called 'sixth sense' (Baker, 1981)) is still very much open to debate. However, there is no doubt that technology greatly enhances man's ability to navigate.

Until the advent of satellites and computing technology, the tools used to aid navigation had changed little for hundreds of years. These tools, the paper map, compass, watch etc. are likely to remain significant aids for the foreseeable future; however, they can be problematic for the mountaineer or hillwalker.

The map and compass do not lend themselves well to the poor or severe weather conditions that the modern user may find themselves encountering. Paper maps deteriorate very rapidly in the rain and are susceptible to every-day wear and tear of being folded and refolded. They are also difficult to use in even moderate winds due to their awkward size and require a light source if being read in poor visibility or at night. There is also the danger that compass readings can be misread or inaccurately transferred onto the map, especially if weather conditions are poor or the mountaineer fatigued or distressed.

The methods used to navigate have always been tied very closely to the tools available at the time. When no maps or compasses existed, the stars and sun would have been the main aids to navigation. At present, the map and compass are the navigator's most popular tools, being cheap and fairly easy to use with only a little background training.

The advent of computers, and satellite technology in the shape of GPS, have revolutionised land navigation. The methods involved when using GPS technology are the simplest yet devised and will increase in use as the technology becomes more widely available. It does have its drawbacks. Even with such advanced technology, the navigator still has to rely on the paper map in order to navigate successfully, as GPS on its own cannot relate the navigator to his or her surroundings.

Using current technology, a device could be created whereby a GPS receiver and an electronic compass could be integrated with a computer, which would display a map in a digital or electronic format. Such a device would provide the modern navigator with the most comprehensive piece of navigation equipment yet, as it would incorporate all the most important elements the navigator requires into one single unit. The features of current navigation activities which would have to be translated into an electronic equivalent are examined in Chapter 2.

To summarise, the main points that can be drawn from this chapter are:

- Map and compass are the most common aids to land navigation.
- The map and compass are cheap and easy to use but their effectiveness can be compromised by poor visibility, weather, or incorrect usage.
- Computing and satellite technology are revolutionising navigation.
- Combining satellite and computer technology with digital maps and an electronic compass would provide the navigator with their most comprehensive and advanced piece of equipment yet devised.
- Such a combination would solve many of the problems that the land navigator encounters with current equipment.

CHAPTER 2: DESIGN CONCEPTS

2.1 The Basis for Design

The downward spiralling costs of computer and satellite technology, combined with the increased availability of digital map data, means that these elements could be combined to create a personal navigation device (Figure 2.1). This system would utilise cutting edge technology to supersede current techniques and navigation aids in their present form, with their inherent deficiencies. The aims of this project were to investigate, to design and to develop a prototype of the navigation software that would be required by such a device.

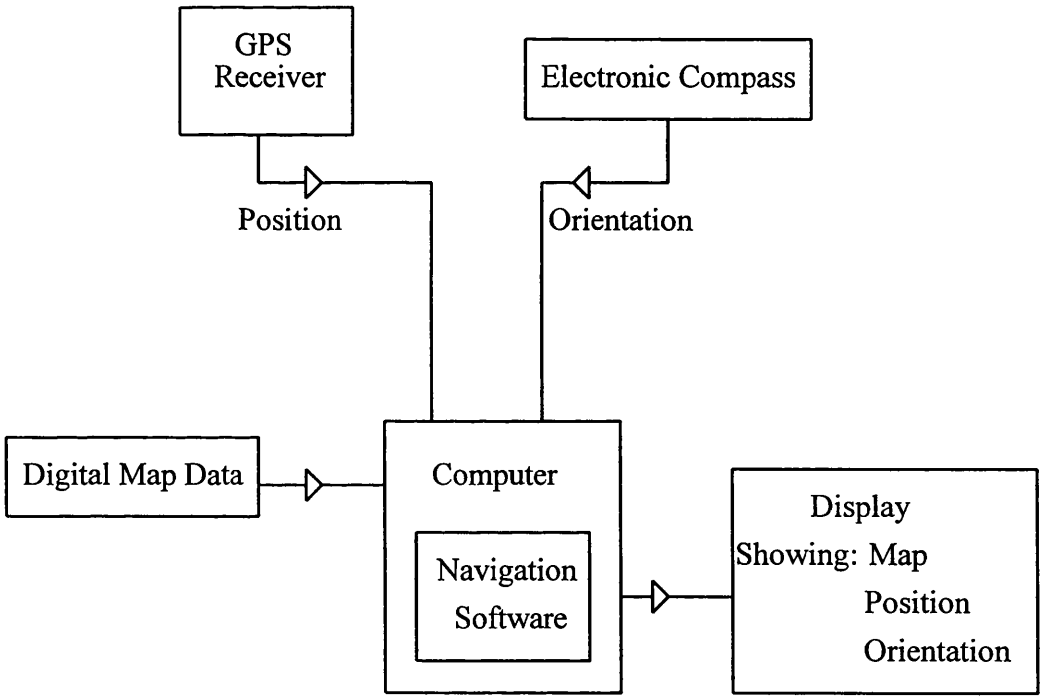


Figure 2.1 Personal Navigation Device

If such software is to be investigated designed and developed, it is important to define the user's characteristics. This follows the ideas of Shneiderman (1987) who stated that:

"All design should begin with an understanding of the intended users..."

The first stage in the design process was to adopt a 'Tools' approach as defined by Bodker (1991) and investigate all the tasks the navigator carries out. These tasks had then to be

successfully translated into the electronic environment, to create an Electronic Navigation Environment.

2.2 The Navigation Process

The typical land navigator performs a series of procedures and functions. He or she must first purchase or select the relevant map sheet for the area of operation. The navigator will choose a map of particular scale, and type, depending on personal preference and the information the map displays. If the map is unfamiliar to the navigator, an inspection of the the legend and marginal notes can be made, to gain some knowledge of how the information is represented on the map, and what co-ordinate and projection systems the map employs.

The navigator will then examine the map on a general level to gain an impression of the area, and to establish the start and end of the journey. The area of the map between these points will then be examined in more detail, either by looking more closely by eye or using a magnifier. This enables the navigator to pick out the detail of the intended area of navigation and establish which areas or features should be avoided and which would allow a more favourable route. This process enables the navigator to plan a route on the map and pin-point specific destinations. Routes can then be measured or scaled from the map to establish distances to be travelled and calculate journey times. This information may then be marked on the map using a pencil and compiled into a table drawn up on paper (known as a route card) for easy reference during the journey.

Once the route has been planned and marked out in this fashion, the navigator must then orientate himself and the map to his surroundings. This can be achieved either by compass, or by determining the navigator's position on the map in relation to the surrounding terrain. The journey will then commence, with the navigator checking his or her position and orientation using the methods outlined in Section 1.3.

The frequency at which the navigator checks his or her route and position will depend on many factors, including experience, personal preference, weather, terrain conditions and speed of progress. If the weather is bad and visibility poor then the navigator may check his or her position frequently to ensure that the correct route is being followed. If weather is good and the journey is progressing faster than anticipated then the navigator may check position less frequently or alter the route or recalculate the journey time. The nature of the terrain will also affect the navigator's checks on position and orientation. If the terrain is hazardous, then position and orientation will be checked frequently because deviation from

the correct path as stated by Tippet (1995) can be life threatening. Barren or featureless terrain will also require frequent checks on position and orientation as it is simple to lose a sense of direction in such terrain.
These processes will be repeated until the navigator reaches the end point of their journey.
To summarise, the navigator performs the following tasks:

- Select map of appropriate area and scale.
- Inspect the map to gain a general impression of the area and determine area of journey.
- Inspect journey area closely to establish journey start and end, as well as hazards to be avoided and places to be visited.
- Mark desired route on map, along with points of interest, areas to be visited.
- Calculate or estimate journey distances, times.
- Orientate oneself with surroundings.
- Undertake journey, checking position, orientation and progress depending on weather, terrain, visibility etc.
- Repeat above processes until journey's end.

This list established the functions and tasks which the navigator would carry out in the course of a journey. For navigation software to be successful, these functions had to be translated into an electronic format; other functions could be added, made possible by the electronic medium, but those functions identified above were the minimum required.

2.2 The Electronic Navigation Environment Concept

At its most basic level the Electronic Navigation Environment had to provide a map display, as all the user functions identified required a map as an essential piece of equipment. A series of tools and commands had to be provided to allow the user to perform the navigational tasks outlined above.

Data storage and retrieval would allow the user to display digital maps. These functions were the digital equivalent of the user selecting paper maps. Map set up functions would allow the user to register new maps and determine map co-ordinate systems. A parallel could be drawn between these functions and the traditional navigator reading the co-ordinate and projection information from a paper map legend.

Map reading requires the user to have full control over the map display, and be able to fully manipulate the display in terms of zoom and pan controls. This required tools that would allow the user to move the map and to magnify or reduce certain areas of the map for

interpretation and inspection. These functions would replace the navigator's need for a magnifier, or to peer more closely at the map. Instead, they could simply zoom into the area of interest.

The navigation environment had to provide some provision for route planning. This included the ability to mark routes and waypoints on the map, and to be able to measure distances. Tools had also to be provided to allow the user to calculate journey times.

Orientation currently requires a map and compass. Although orientation can be achieved without a compass, it is still the most effective and widely used method to orientate oneself and therefore would be included in the software, in the form of both a graphical and numerical display.

Position Determination requires a map display and GPS to allow the system to operate 24 hours a day. A digital map and electronic compass would be of little use at night for determination of position as no physical features can be seen (see Section 1.3). GPS was seen as an essential requirement. The position gained from the GPS receiver would be displayed on the digital map. This process would be carried out by a piece of software known as a GPS Driver.

Finally, as recommended by Sutcliffe (1988), a suitable help system had to be provided as an on-line manual to aide the user in case of difficulty when using the software.

All this information had to be conveyed to the user in a simple and effective manner. This required the incorporation into the navigation environment of a suitable Software User Interface. As will be seen from the User Profile (Section 3.2.2), the system was not to be designed in the expectation that it would be used regularly, and therefore complex or difficult to learn user interfaces were to be avoided. The Electronic Navigation Environment therefore required an interface that would be quick to use and simple to learn. To summarise, the Electronic Navigation Environment had to contain the following elements (Figure 2.2):

- Map Display
- GPS Driver
- Tools
- Software User Interface

The main purpose of the Electronic Navigation Environment was to provide the user with their location on a digital map. With this in mind the software was to operate as follows (Figure 2.2).

The positional data required would be provided by a GPS receiver. This information would be transferred by the GPS Driver to the Map Display.

The user would then be able to manipulate the displayed data, to zoom and pan around the digital map as desired, and store or plot route information. The user would interact with the software through a suitable user interface.

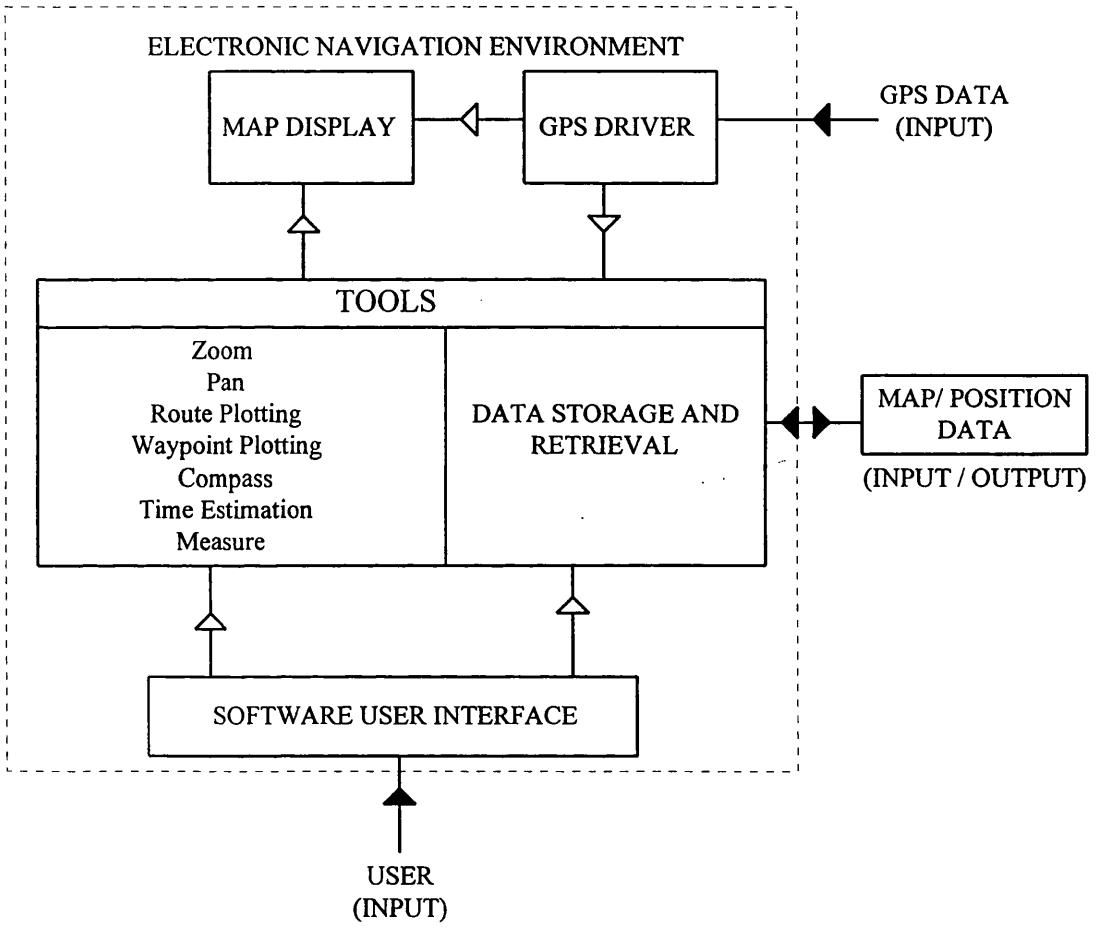


Figure 2.2 The Electronic Navigation Environment Design Concept

With a conceptual model of the software established, the methods and equipment required to develop the Electronic Navigation Environment concept further had to be addressed.

CHAPTER 3: METHODOLOGY

3.1 Design and Development Methods

This chapter discusses and explains the methods used to design and develop the Electronic Navigation Environment. The chapter describes the methods used in designing the software from conceptual design to program planning. Coding and debugging techniques are also discussed, before the methods used to design the Software User Interface are investigated. The methods used during the investigation can be divided into two sections: designing and developing. Different design techniques were employed at different stages of the investigation. Developing the navigation environment involved the use of a structured approach to coding.

3.2 Design Methods

A Top-Down approach was adopted for the overall design of the software. As discussed by King (1988) and Cassel (1983) this involved starting with an abstract concept of the tasks the program will perform as well as how it will operate, then gradually refining this concept down to individual procedures, processes and code. However, as will be discussed in this chapter, other methods were incorporated into the Top-Down approach to make it as flexible as possible.

The following design stages were used:

- Conceptual Design
- User Profile
- Software Design
- Software User Interface Design

3.2.1 Conceptual Design

As discussed by Cassel (1983) conceptual design of the software was required to give the designer a basis on which to build a more detailed proposal. This involved defining the requirements of the software in general terms (as identified in 2.1), and the operation of the software, in direct relation to the user. This was achieved using the 'Tools' approach discussed by Bodker (1991), which is based on the idea that "new technological designs are

developed as alternatives to existing technology and tools". The conceptual design provided the designer with a basis on which to select the relevant equipment and software for development, as well as leading to a more detailed design specification.

3.2.2 User Profile

The User Profile was essentially the designer's idea of who the user would be and what tasks they would perform. The User Profile was established by building up an outline of the User's characteristics (as discussed by Sutcliffe (1995) and Booth (1989)) which are explained below.

- User Definition: Provides a short description of the typical user:

  User Definition: Mountaineer or Hillwalker.
  The definition of the user was a mountaineer or hillwalker. These two terms are often used interchangeably; however, the mountaineer is often seen as being more experienced and walking in more extreme environments than the hillwalker.

- Computer Familiarity: How computer literate the user is, i.e. Naive, Novice, Skilled, Expert:

  Computer Familiarity: Variable.
  This factor was deemed varied, due to the wide range of people who are involved in mountaineering. Many mountaineers will use computers in every day life and so will be computer literate and familiar with software use. Some will have little experience with computers and software and so the system must reflect this varied range of computer familiarity.

- Discretionary Usage: Whether the system is used voluntarily or compulsory:

  Discretionary Usage: Voluntary.
  The use of the software was seen as predominantly recreational, and therefore the user employs it voluntarily. It may also be used by mountain rescue and emergency services, whereby its use would become compulsory.

- Frequency of Use: How often the user will use the system. For instance if the system is to be used frequently then the user interface can be complex. If the system is to be used irregularly the interface should be simple, quick to learn, and easy to remember:

Frequency of Use: Irregular.
The recreational nature of the software's use means that it will generally be used infrequently (with the exception of mountain rescue and emergency services).

- User Knowledge: The level of knowledge the user has of the system and its purposes/ functions:

User Knowledge: Varied.
The user's background knowledge will depend on their level of theoretical and practical land navigation. The more experienced the mountaineer is in a navigational sense, the better idea they will have of what the software is supposed to do and what functions it will offer.

- General Abilities: The general knowledge and intelligence of the user, the level of background knowledge the user has of the program applications:

General Abilities: Varied.
The actual navigational experience of mountaineers varies widely, from those with sound practical and theoretical knowledge, to those who rely on memory and little else. A recent study carried out on the Scottish Mountain Rescue by Anderson (1995) revealed that out of all the casualties between 1989 and 1993, 22% carried no navigation equipment, and of those that did, 39% were unable to use it. This contrasts with many mountaineers who are proficient and very experienced with current navigational techniques.

- Physical Abilities: The physical characteristics of the user and the environment in which the user will utilise the system. This aspect will relate to the ergonomics of the system.

Physical Abilities: Mountain environment, harsh terrain and all weather conditions.
The environment in which the user will operate is outdoors, usually in the harsh weather conditions and rough terrain of mountain regions. This means operation in all weathers, day or night.

- User Functions: The type of tasks / goals to be achieved through using the software package:

User Functions : This section summarises the type of tasks and functions the mountaineer would typically perform when navigating (Table 3.1).

Task	Equipment
Determining Position	Map & Compass, Map, GPS
Determining Orientation	Map, Compass
Route Planning	Map, Compass, Measuring Device
Route Following	Map, GPS
Map Reading	Map
Time/ Distance estimation	Map, Measuring Device, Watch

Table 3.1 User Functions

The User Profile method was originally developed for the design of user interfaces, but was equally appropriate when applied to the design of the Electronic Navigation Environment software as a whole. The User Profile defined above was used as a basis for creating the SUI, GPS Driver, Map Display and Tools.

3.2.3 Software Design

The design phase is an essential process that should be undertaken before program development. As discussed by McGowan and Kelly (1975), it was essential to adopt a structured approach to planning and designing the program otherwise there was a danger of losing sight of the original aims of the program. A Top-down method was adopted for the program design phase, as it allowed the design to be broken down into small manageable modules.

Top-down design allows the user "...to identify the major functions to be accomplished, and then to proceed from there to an identification of the lesser functions that derive from the major ones" Yourdon (1975).

This seemed the most logical approach to program design, working from the whole to the part (Figure 3.1).

- Program Aims
- Program Operation
- Flow Diagram
- Step by step descriptions of program procedures using pseudo code
- Establish code required to create above procedures

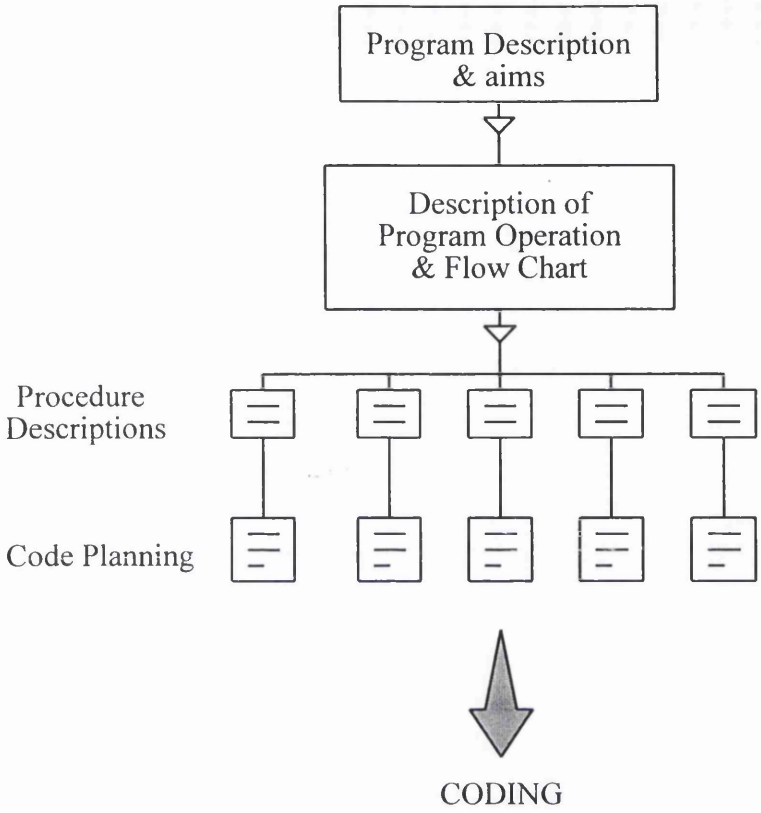


Figure 3.1 Top-Down Design

The plan began by stating very simply what the program was intended to do, followed by a paragraph on how it was envisaged that the program would operate. This allowed the programmer to focus on the problems to be solved, and could be referred to throughout development. Failure to carry this process out could have lead to the program deviating from its original requirements or becoming bogged down with a specific problem, when a glance at the purpose of the program would have provided a simple solution. The next stage in planning the program was to prepare graphically how the program would operate.

These steps in planning the program were created to state the aims and general operation of the program, as well as establishing how any data involved in the program would behave.

The next stage was to write out the procedures that were required, and the function each would perform. This was carried out by writing in pseudo code as defined by LaBudde (1987) and Cassel (1983).

This process served a similar purpose to that described above, but operated at a lower level. Once again, these plans could be referred to throughout development.

The next stage of planning was to work out what code was required to achieve the objectives of each procedure, to establish how any loops were going to operate, and how

they would end. Once this had been established then the programmer could proceed to coding.

Other design methodologies were investigated and applied where appropriate. Areas of the program that were dependant on the input, flow and output of data were designed using the Jackson Methodology, as defined by King (1988). This design method is based on the theory that the design and construction of a program is shaped by the data it handles. Furthermore, it adopts the view taken by Dijkstra (1965) that any program consists of only three elements; sequence, selection and iteration.
The Jackson Methodology is different to the Top-Down approach, which is function and process based in its approach. However, despite these differences, it is stated by King (1988) that both methods can be used to complement each other.

3.2.4 Software User Interface Design

The design of the Software User Interface required the use of specialist methods and techniques in order to create the most efficient and user friendly interface possible. Although the user interface is a piece of software, the design methods discussed previously were not appropriate.
In order to successfully design the Software User Interface, the choice of design methods was vital. The design of the user interface relied heavily on the User Profile. From the User Profile, the System Image could be designed (Figure 3.2). This was essentially what the user would see and interact with when using the ENE.

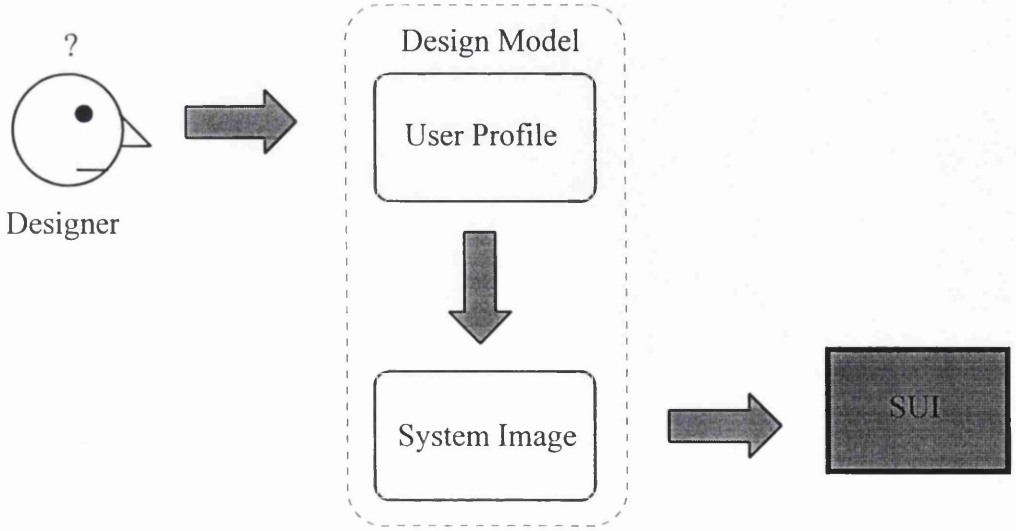


Figure 3.2 Design Model

The System Image was designed using method guidelines outlined by Sutcliffe (1995). These are discussed below.

(i) Command Naming
(ii) Menu System
(iii) Representation of Options
(iv) Colour
(v) Help Prompts

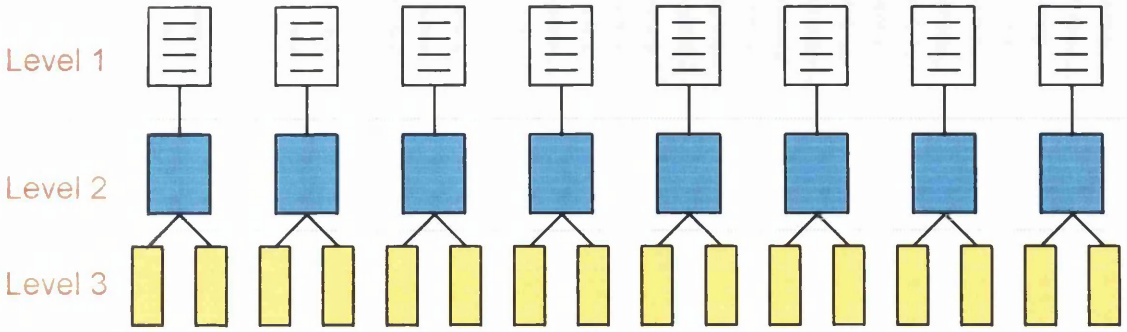
(i) Command Naming

Command Naming dealt with what text would be used to represent the various options outlined in the Conceptual Design of the program. Shneiderman (1987) identified the fact that care had to be taken to chose the correct words to avoid confusion and misinterpretation.

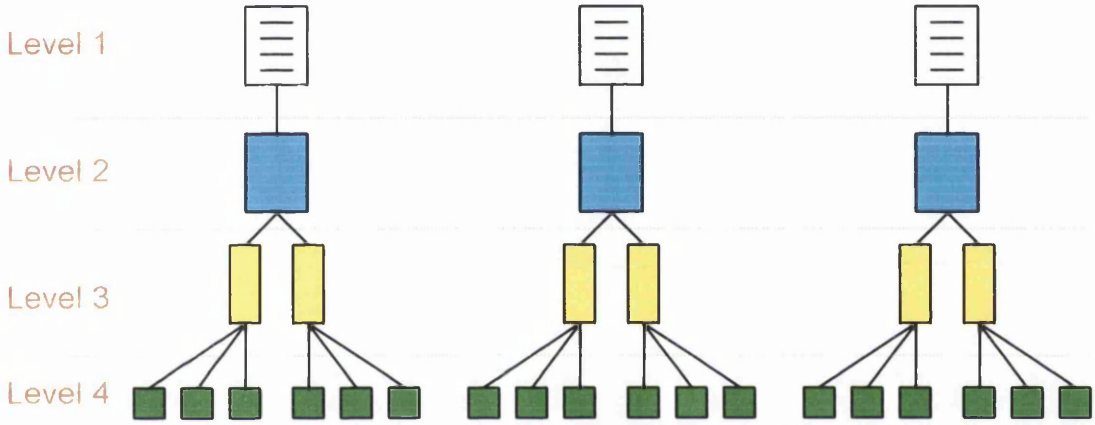
(ii) Menu System

As stated by Shneiderman (1987) a Menu System can be defined by its breadth and depth (Figure 3.3). The deeper a menu is the more options the user must select to reach their goal, but the user has fewer visible options to choose from and so the interface appears simple. However, below level one it may prove to be a complex web. A wider, more shallow menu takes less time to reach low level options but presents more visible options to the user. This could lead to confusion, but the response time to reach lower level functions is quicker.

The correct balance had to be found between the menu breadth and depth, in order to achieve maximum clarity and minimum response time. In order to establish this balance a function hierarchy would be created. Sutcliffe (1988) states that the hierarchy can group common functions together, as well as ordering them in terms of importance to the user and frequency of use. A function hierarchy could be achieved by many means, including questionnaire, designer discretion or using a more statistical approach.



This menu has an 8x3 format. This means it presents the user with eight visible options (Level 1). In order to reach the bottom of the menu system the user must go through 3 levels.



This menu is a 3x4 menu. The user is presented with fewer visible options but must travel deeper and through more levels in order to reach Level 4 functions / options.

Figure 3.3 Menu Breadth and Depth

(iii) The Representation of Options

The Representation of Options dealt with how each option or function would be conveyed to the user. e.g. Command Language, Natural Language or use of a Graphical User Interface (GUI). Sutcliffe (1988) states that Command Languages provide the user with a set of command words (known as a lexicon). In order for the user to utilise a program function, the appropriate command word is typed via a keyboard into the computer. Command languages are the oldest form of Software User Interface, and examples of this include Microsoft Disk Operating System. These languages are sophisticated and flexible and are ideally suited to complex programs with many functions. Shneiderman (1987) states that the drawbacks of command languages are that they can be difficult and time consuming to learn. The user has little on-screen information to aid in computer interaction, and must have a comprehensive idea of the program's purpose and functionality.

Natural Language interfaces as defined by Sutcliffe (1988) rely on the user typing in what the user wants the program to do, in the form of written English. This is the ideal interface

as it requires no learning from the user, as long as they can write. However, the computer must interpret what the user has written in order for the program to perform. This is extremely difficult due to the expansive nature of human languages and the differing meanings of words depending on their contexts.

The final type of Software User Interface is the Graphical User Interface (GUI). This relies on a series of menus, icons and messages in order to allow interaction (Figure 3.4). Basically, the GUI presents the user with a series of choices that can be selected in order to make the program perform its functions. As discussed by Shneiderman (1987) the GUI does not require the user to remember any command words, but to associate the relevant menus and icons to specific functions. Shneiderman also notes that this type of interface is quick to memorise and easy to learn. However, compared to command languages, a GUI is fairly inflexible and restrictive.

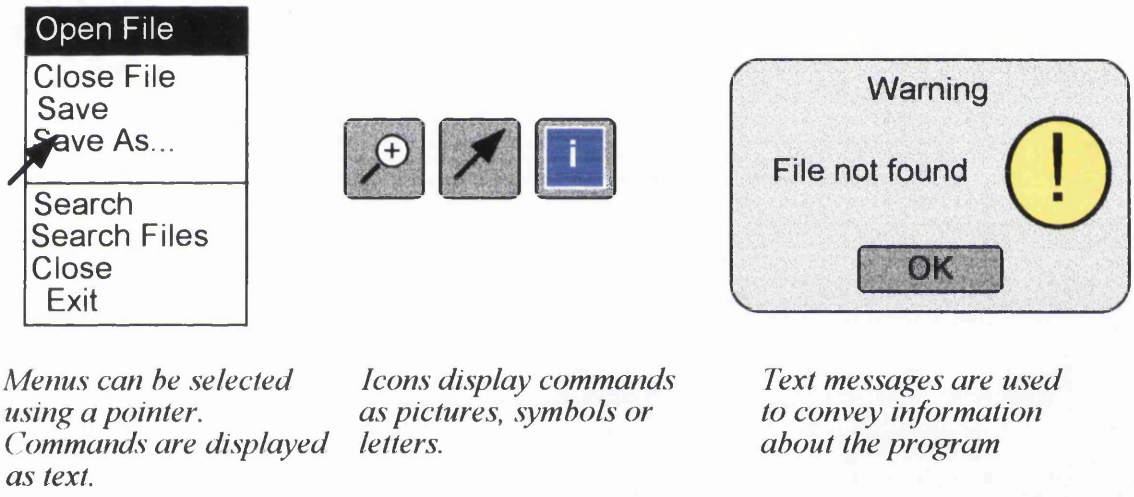


Figure 3.4 Graphical User Interface Representation

(iv) Colour

The colours used in the SUI are extremely important as they serve to highlight the more important aspects of the system. Colour can also make the SUI appear more aesthetically pleasing. From the texts by Travis (1991), Sutcliffe (1988) and Shneiderman (1987), a series of guidelines for the use of colour were laid down:

1) Use colour conservatively; limit the number of colours used to 6-7. Too many colours can create confusion or garish displays. The fewer colours used, the easier it is for the user to accept them.

2) Use colour coding and association (e.g. red symbolising danger, or yellow depicting a warning).

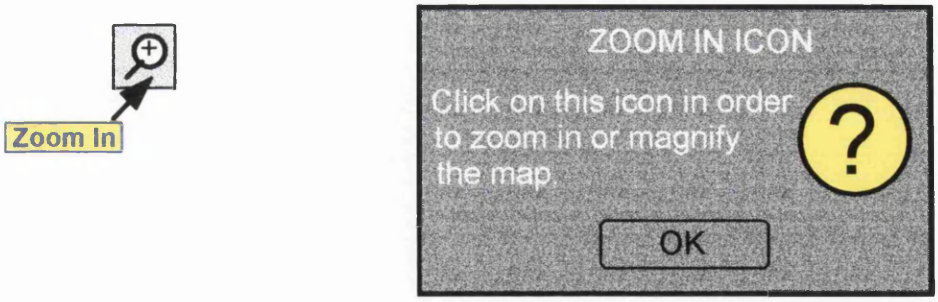
3) Use colour to highlight and emphasise.

4) Choose colours that can be easily discriminated from each other.

The selection of colour was dependant on hue (spectral characteristics), value (perceived lightness or darkness of a colour) and saturation (purity of the colour). The appropriate selection was vital to provide a clear, visually attractive SUI. The use of the wrong colours could cause confusion or appear unfriendly or disturbing to the eye, and thus repel the user.

(v) Help Prompts

If the system is complex the user may require help from the system in order to carry out certain functions or commands. A less complex system will require fewer help prompts. How the help prompts are represented depends on how much information needs to be shown, e.g. simple help prompts may be bubble text, more complex help may be dialogue boxes (Figure 3.5).



Help Bubbles provide a quick, concise way of providing User help.

Dialogue Boxes allow more detailed help information to be displayed.

Figure 3.5 Help Prompts

These are the basic concepts and methods that were used to design the SUI. This approach, if used correctly, would ensure that the design required the minimum of redesign and revision.

3.3 Development Methods

Once the design process had been completed, then development could proceed. Software development can be divided into two stages:

- Coding
- Testing and Debugging

3.3.1 Coding

Three coding methods were used during the development of the software: Top-Down coding, Bottom-Up coding and Modular coding.
Top-Down coding was employed for creating menu systems as McGowan and Kelly (1975) state that Top-Down coding is a hierarchical method which naturally follows the pattern of the menu system.
The Bottom-Up coding method was used where the program had to operate in real-time. This allowed the code to be written and tested as it was built up to ensure that each function or statement operated before a new one was added. A modular approach could not be adopted as this is not always practical when developing programs that operate in real-time (Yourdon 1975). According to LaBudde (1987) individual modules could be slow to interact with each other and so the system would not operate in real-time.
A modular approach, as defined by LaBudde (1987) and Brown and Sampson (1973), was used where common elements of the program that were independent of each other were to be coded. These elements could be tested individually when created in a modular fashion until operational, then added to the program. The Top-Down and Bottom-Up coding methods would require that these elements were created in parallel, so that testing could not take place until all elements had been fully created. This would cause a build up of errors which, as stated by LaBudde (1987) ,the modular approach reduces.

Each procedure or piece of code was annotated with comments, as, according to Gurewich and Gurewich (1995) if the code needed to be revised or debugged, the comments would provide guides as to the nature and purpose of the code. Code written without comments could have been extremely difficult to follow, complicating revisions and debugging.

3.3.2 Testing and Debugging

The overall testing method that was adopted to ensure the code, procedures and system functioned was that defined by Abbot (1986). This method involves three major steps:

- Module Testing
- Integration Testing
- System Testing

Module Testing ensured each procedure could operate in isolation. Integration Testing was used to ensure procedures interfaced successfully with each other. Finally, System Testing was employed to evaluate the application as a whole (Figure 3.6).

The advantage of this method was that it would reduce debugging time as each procedure operated correctly before being run with other procedures. The drawback was that although individual procedures might run by themselves, they could be affected by actions taken by other procedures in the program, and problems arising could prove difficult to find when each procedure was treated as an individual block.

The alternative was to write the program en masse, then test and de-bug. This method had obvious shortfalls in that errors made during writing would be accumulated, which would lead to a very lengthy de-bugging process.

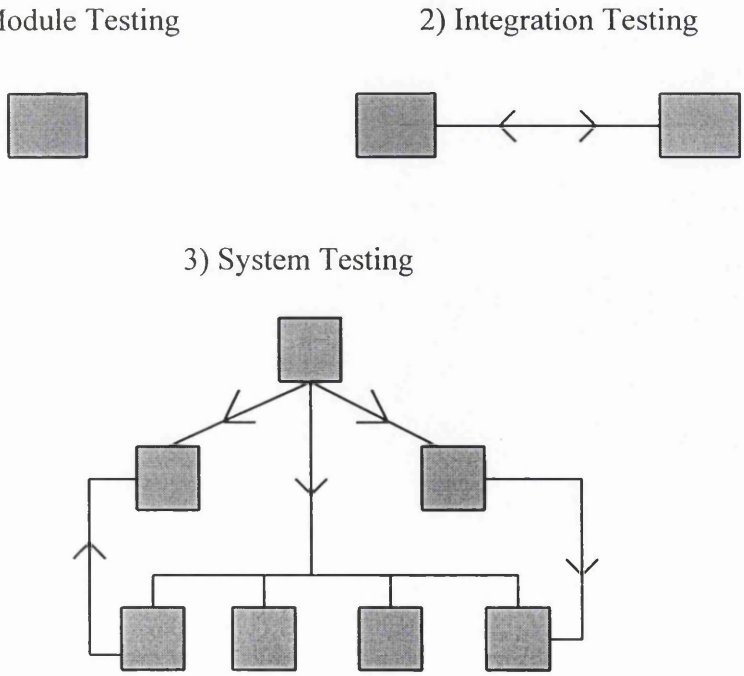


Figure 3.6 Testing Methods

Within this overall testing method, many techniques were applied to test and debug different aspects of the program. Different tests were applied depending on the nature and structure of the code being tested. These test methods were seen as part of the coding and assessment process and are defined and discussed in Chapter 6 and 7, together with the reasons for implementation.

3.4 Summary

This chapter has outlined and explained the methods and techniques employed to design and develop the Electronic Navigation Environment. It is clear from the methods chosen that a structured approach was selected for each stage in the design process, with definite step by step procedures employed throughout.

At first this may appear to be a regimented and blinkered approach, but many of the methods used either had flexibility built in or had been tailored to suit the task. One example of this was the ability of the Top-Down process to accommodate the modular and Jackson methodologies.

The testing methods employed have been discussed on a general level in this chapter, as it was felt that a more detailed and justified account of the methods used could be given where and when each method was employed.

Overall the methods and techniques chosen were felt to be the best suited to designing and developing the Electronic Navigation Environment, and not necessarily those that were easiest to use or those currently most favoured in software design and development.

CHAPTER 4: EQUIPMENT AND SOFTWARE SELECTION

4.1 Equipment and Software Requirements

The equipment and software required to design and code the navigation environment were based on the conceptual design and methods set out in chapters two and three. The design and development of the Electronic Navigation Environment was divided into four sections:

(i) Map Display
(ii) GPS Driver
(iii) Software User Interface
(iv) Tools

In order to design and develop these sections, hardware and software were required. The following general groups of equipment were required for each stage (Table 4.1).

Stage	Equipment Required
i	Computer, Mapping Software, Development Language(s), map data
ii	Computer, Mapping Software, GPS & SDK*, Development Language(s), map data
iii	Computer, Development Language(s), Graphics Tools
iv	Computer, Mapping Software, Development Language(s), map data, Electronic Compass

** SDK = Software Development Kit*

Table 4.1 Equipment Types Required

From Table 4.1 it is evident that the equipment required could be divided into items of software, hardware, and map data:

Software	Hardware	Map Data
Mapping Software	Computer	Digital Maps of Great Britain
Development Language(s)	GPS Receiver	
SDK	Electronic Compass	
Graphics Packages		

The requirements for these items were investigated, followed by an assessment of the various options available. Finally, a suitable product was selected for each of the above categories.

4.2 Software

The following software was investigated and selected for Electronic Navigation Environment development, testing and analysis. The software was selected first, as this dictates the type and specifications of the hardware.

4.2.1 Mapping Software

The software that was to be employed as the map display had a number of criteria which it had to fulfil in order to be a suitable candidate. It had to be able to display the appropriate map data in full colour, and recognise coordinate and projection systems so that the GPS positional information could be related to the map data. The software had to be fully developable so that it could be tailored to fit the application. Finally, the data had to be organised into layers, so that data could be added and removed to and from the map without any adverse effects. The requirements for the map display software are summarised below:

- Display full colour raster and vector map data in a wide range of formats
- Fully developable so that the package can be customised
- Able to recognise coordinate and projection systems
- Contain layers for easy addition/deletion of data
- Capable of storing and displaying route and positional information
- Capable of receiving and manipulating GPS data
- Able to integrate or communicate with a GPS receiver

Several types of software were considered for use as the map display. These included:

- Geographical Information Systems
- Graphics Packages
- Computer Aided Design Packages
- Development Languages

Parent (1988) defines Geographical Information System (GIS) as "...a system that contains spatially referenced data that can be analysed and converted to information for a specific set of purposes or applications...". They are designed primarily to allow the display, management and analysis of spatial data (Antenucci *et al*, 1991). A typical GIS will contain three main elements:

- Cartographic functionality - the ability to create, store and edit maps and plans
- Analytical functionality - allows processing and complex investigation of spatial data
- Data Management - enables efficient storage and retrieval of data

GIS systems are designed to be flexible so they can be tailored to suit the needs of the user and nearly all can be modified or developed. They can accept a wide range of map data, and are often designed to communicate with other packages, especially databases and spreadsheets. Examples of GIS systems include MapInfo, Atlas and ARC/INFO.

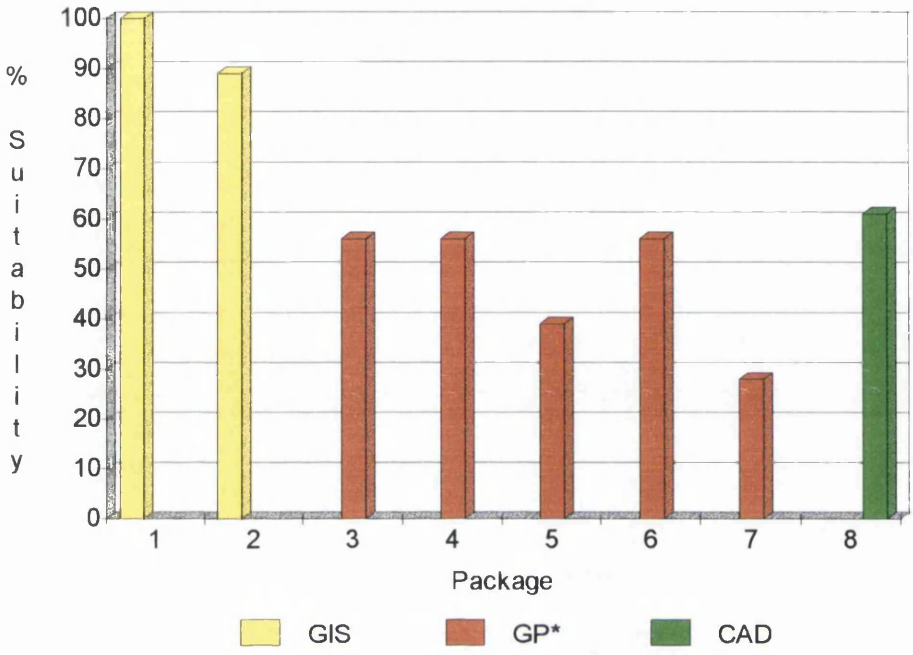
Graphics packages are used primarily to design graphical images such as illustrations or sprites for computer animation. They provide functions for viewing graphical data, as well as tools for editing existing images or creating new images. Examples of such packages include Paintbrush, PaintShop Pro., and MapViewer. The complexity and range of functions available on these packages varies greatly, as does the range of data types they can accept. They are primarily designed to deal with raster data.

Computer Aided Design (CAD) packages are primarily used in engineering, architecture and surveying to produce plans and complex line drawings. CAD packages vary greatly in functionality and design, but all possess the same basic functions, such as an understanding of coordinate systems, the ability to manipulate drawings and plans, and the relevant tools to allow the creation of new drawings or editing of existing drawings. The data that can be accepted by CAD packages varies greatly, but they are predominantly designed to accept vector data. Examples of such packages include AutoCad, TurboCAD and FastCAD.

Finally, development packages could also be used to create a mapping system. These include systems such as C++, Visual Basic, and Delphi, that provide the tools to create highly visual programs with the minimum of code. However, although these would have provided the best solution to creating a map display for the Electronic Navigation Environment, it would have been an extremely complex and lengthy process. A complete map display package would have had to be developed, involving the creation of a graphics viewer and a coordinate and projection system interpreter, as well as the facility to edit and manipulate the map data. This approach, therefore, was deemed too complex and time consuming, and so it was decided that an off-the-shelf package would be used to provide a design platform, which could then be modified to suit the requirements of the navigation system.

Once development languages were eliminated from the investigation, information on a selection of the three remaining types of software was obtained, together with

demonstrations of the software's capability. From these investigations, the following chart was created to illustrate the suitability of the various packages to the task of displaying and manipulating map data (Figure 4.1).



**GP = Graphics Package*
(The numerals correspond to a specific package. The suitability was calculated using the system outlined in Appendix I).

1. MapInfo v3.0
2. Atlas v3.0
3. Surfer 3D
4. Mapviewer
5. Landscape Explorer
6. Mapvision
7. LView Pro.
8. DeskMapper

Figure 4.1 Suitability of Packages

As can be seen from Figure 4.1, the GIS packages were the most suitable systems for the task. The investigation of the CAD package (AutoDesk Ltd., 1995) revealed that it posssesed limited development capabilities, combined with restricted data acceptance and communications capabilities, meant that the system investigated was too limited. The graphics packages proved to be least compatible. The actual suitability varied considerably by up to 28%, reflecting the range of packages on offer. Ultimately, graphics packages were too restrictive in terms of communications, development and the range of data they could accept to provide a useful map display. This assessment clearly showed that GIS systems were the most suitable packages to be employed.

MapInfo was the GIS system selected as the design platform for the map display, as it already provided many of the functions required in terms of map manipulation and editing. It was selected over similar GIS systems for a number of reasons.

MapInfo is a relatively small and compact system compared to the more traditional GIS systems such as ARC/INFO. The MapInfo GIS is designed to be run and used on a personal computer whereas a system like ARC/INFO is designed for use on a high-end workstation. Compared with GIS systems of a similar scope and focus (e.g. Atlas GIS), MapInfo has the advantage of having moderate memory requirements and better functionality in terms of development potential, map data and communications (Reeve, 1996) (Table 4.2). Finally, MapInfo also achieves 100% compatibility to the task, compared to Atlas with 89% (see Appendix I).

	Mapinfo	**Atlas**
Min. Processor	286	386
Memory (Mb)	12	20
RAM (Mb)	8	8
Development	MapBasic	Script
Raster Data	All Major Formats	Limited Formats
Vector Data	All Major Formats	All Major Formats

Table 4.2 MapInfo vs Atlas
Source: Mapinfo Corporation (1995) and Strategic Mapping Inc. (1995)

4.2.2 Development Languages

The development languages to be used depended primarily on the software chosen to provide the map display. MapInfo was selected to provide the map display, and is capable of working in conjunction with three development languages.

(i) MapBasic
(ii) C++
(iii) Visual Basic

(i) MapBasic

MapBasic is the development language designed for use with MapInfo. It is based on Visual Basic but has been tailored to suit the functions and applications that are required by MapInfo. It has a fairly limited array of commands, making it easy to learn and simple to use, but at the same time many tasks are beyond MapBasic's scope. For this reason it is

capable of running applications written in C++ or Visual Basic, in order to increase its flexibility.

(ii) C++

C++ was designed as a complex programming language capable of being used for practically any task. Because of this, it is an extremely flexible language, but this in turn means that it is complex and difficult to learn. Visual C++ was designed to provide a system capable of creating Windows-based programs by providing the tools to create menus and Graphical User Interfaces using an extensive range of tools similar to those found in graphics or CAD packages. However, the language is still as complex and difficult to learn as C++.

(iii) Visual Basic

Visual Basic was designed to be used along the same lines as Visual C++. However, the language itself is much easier to learn but at the same time does not provide the same flexibility as Visual C++.

The languages chosen for program development were MapBasic and Visual Basic. MapBasic was an obvious choice as it is designed to develop MapInfo and can modify the system itself. Visual Basic was chosen due to its similarity to Map Basic, and the functionality it provides was deemed suitable for GPS Driver and Software User Interface (SUI) development. C++ could have been used, but its complexity and increased functionality was not required for SUI design or, as will be seen later, GPS driver development.

4.2.3 Software Development Kits

A Software Development Kit was required for GPS Driver development. Its main use would be to provide an interface between the GPS receiver and the map display software. All the development kits investigated were specified as being capable of this, and so were investigated on the strength of their flexibility, user friendliness and documentation provided. With only three GPS receivers currently on the market in the UK with Software Development Kits, all were investigated. These were:

(i) Silva Nexus
(ii) Trimble Software Development Kit
(iii) Rockwell Software Development Kit

(i) Silva Nexus

This package is used in conjunction with the Silva GPS Compass and Personal Computer interface cable so that the GPS data can be collected and viewed on a computer. The software can be used to log data for analysis and storage (Silva, 1994a). The Silva Nexus can be modified and developed using Microsoft Quick C and the program source code which is provided with the development kit. Also provided was a document listing the various communication procedures and data packets that can be received from the Silva GPS Compass (Silva, 1994b).

The Silva Nexus proved to be a relatively poor software development kit for a number of reasons. The source code provided for development was in Quick C, which requires a Quick C compiler. Unfortunately Quick C is obsolete and is no longer available, making development impossible. The documentation provided with the development kit was very poor in terms of content. Only listings of data packets and communication procedures were given, and little help was provided on how the code could be modified. The software development kit was DOS based, relying on a command language as a user interface.

(ii) Trimble Software Development Kit

The Trimble Software Development Kit was provided with the Trimble Mobile GPS receiver. The development kit is provided with a programmer's guide and series of commands that can be used with both C++ and Visual Basic languages. The commands allow data to be read from and sent to the GPS receiver. The development kit is also provided with a series of program examples in both Visual Basic and C++ that can be modified by the programmer to suit their own needs.

This software development kit proved to be extremely user friendly, with good documentation both for the advanced developer and novice. The fact that more than one language could be used gave the development kit flexibility, and the many programming examples meant that it was very user friendly.

(iii) Rockwell Software Development Kit

The Rockwell Software Development Kit was designed for use with the Rockwell NavCard. It was the most advanced development kit, providing source code in C++ and Visual Basic, as well as receiver monitoring programs and utilities to integrate navigation data into programs (Telecom Design Communications, 1995). Although the kit was extensive and allowed the user to modify the GPS receiver in practically every way possible, the software itself was difficult to use. The documentation provided and help available was fairly poor, although a copy of the actual manual could not be obtained for this investigation.

The Trimble Software Development Kit proved to be the most attractive package to use, due to its user friendly nature, program examples and good documentation. It clearly showed that this development had the potential to integrate the GPS receiver with MapInfo, and this selection was extremely important when considering which GPS receiver to use (see section 4.3.2).

4.2.4 Graphics Tools

Graphics tools were required for a number of reasons. This included creating flowcharts for software design, as well as creating icons and other pictorial displays for the Software User Interface (SUI). Three packages were investigated for these tasks:

(i) Paintbrush v3.0
(ii) PaintShop Pro v3.0
(iii) Microsoft Draw

(i) Paintbrush
Paintbrush is a graphics editor provided with Windows v3.0. It is a relatively simple system, allowing the user to edit existing raster images or create new images using a number of drawing tools and functions. The package's simplicity means that it is easy to use and learn, but restricted in terms of functionality, relying instead on the user's artistic ability to create professional images rather than providing tools that would achieve this.

(ii) PaintShop Pro
PaintShop Pro is a graphics editor similar to Paintbrush, but provides a much more comprehensive series of tools and functions to aid in designing graphical images. It is extremely flexible, accepting a wide range of graphics formats, both raster and vector. However, the increased functionality means that the package is more complex to learn and use.

(iii) Microsoft Draw
Microsoft Draw is designed primarily to produce line drawings and diagrams. It is extremely basic in terms of functionality, but very easy to use. The resultant diagrams can be very professional, but depend heavily on the user's artistic ability.

The packages reviewed for use as graphics tools were all different, providing varying functionality. It was decided that all three packages could be used efficiently, with

PaintShop Pro being used for icon design, due to the professional images that could be created with relative ease. Paintbrush would be used to create simple pictures for use in the SUI, or to create more pictorial diagrams required in the design stage of the project. Microsoft Draw would be used to create flow charts and simple design specifications, as well as producing the guides for SUI design.

4.3 Hardware

With the appropriate software selected for the investigation, the relevant hardware was then selected.

4.3.1 Computer

The type of computer used during the investigation was vitally important as it would be used to design, develop and test the Electronic Navigation Environment. The investigation at its most basic was involved with navigation, which requires mobility. Therefore, the computer had to be fully portable, and capable of operating outdoors. It had to provide a suitable hardware interface to allow the connection of a GPS receiver. The processor and Random Access Memory had to be capable of operating the software efficiently, as well as manipulating the map data used by the map display software. The computer also required sufficient hard disk space to allow storage of both the software required and the map data. Therefore, the computing requirements were:

- Fully Portable
- Sufficient processing power
- Sufficient Random Access Memory
- Large Hard Disc Drive
- Interfaces for GPS Receiver

It was decided that a notebook computer was required to provide a portable design, development and testing platform. The computer selected was a Viglen DX4-100 with 8Mb Random Access Memory, 560Mb hard disc and personal computer card slots type I to III, as well as communications ports. The 100Mhz processor with maths co-processor was required for the large amounts of data that had to be manipulated, as well as the programming to be carried out. The 8Mb Random Access Memory was the recommended value for the MapInfo system, and once again would allow quick processing of data. The large hard disc drive would be used mainly to store digital map data, with the personal computer card slots and communications ports allowing GPS-hardware interfacing. The

whole computer is fully portable, with Nickel Cadmium batteries providing the power source when the unit is mobile.

4.3.2 Global Positioning System Receiver

As already mentioned in section 4.2.3 there are three possible GPS receivers that could be used for the investigation. The selection relied heavily on the choice of software development kit; however, the hardware aspects of the receiver were also considered. The three GPS receivers that could be used were:

(i) Silva GPS Compass
(ii) Trimble Mobile GPS
(iii) Rockwell NavCard

(i) Silva GPS Compass
The Silva GPS Compass is an integrated GPS receiver and electronic compass hand-held unit. The device uses a standard Rockwell Microtracker GPS receiver, accurate to within 100 metres (with selective availability), and has an electronic compass accurate to 1 degree. The unit is capable of storing route and waypoint information, as well as calculating course over ground, bearing to next waypoint, estimated time of arrival and many other pieces of information. The receiver also contains a number of different datums which allows the user to display position in a number of coordinate reference systems. In order to interface the unit with a computer, a personal computer interface cable is required. It has a standard RS-232 interface that connects with the Serial communications port on the computer, and a 4-PIN plug that connects to the GPS receiver. In order to interface the GPS receiver and computer, a 12 volt power supply was required to be connected to the cable (Figure 4.2).

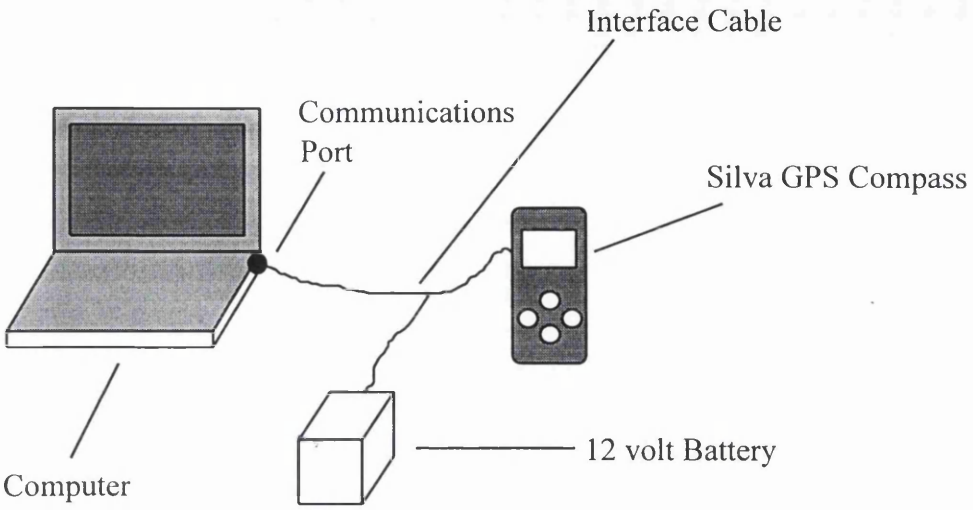


Figure 4.2 Silva GPS - Personal Computer Interfacing

(ii) Trimble Mobile GPS

The Trimble Mobile GPS Receiver is manufactured as a personal computer card. The card houses a SVeeSix receiver accurate to 100m (with Selective Availability), and requires an external antenna, which connects directly to the personal computer card.

The card can easily be interfaced with the computer via the type II interface. The unit is powered by the computer, requiring a 5 volt supply (Figure 4.3).

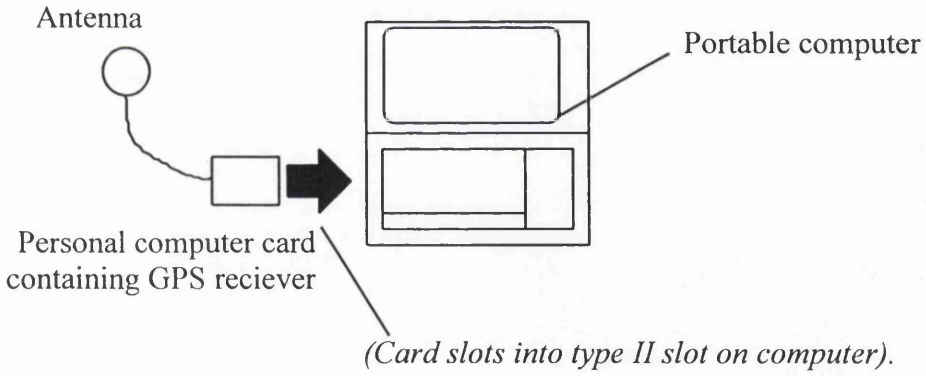


Figure 4.3 Trimble Mobile GPS - Computer Interfacing

(iii) Rockwell NavCard

The Rockwell NavCard is, like the Trimble Mobile GPS, manufactured as a personal computer card. The NavCard specifications are practically the same as the Trimble card, although, as described by Rockwell (1996), the antenna can be docked with the card itself or removed, and be connected via a cable (Figure 4.4). The unit is interfaced with the computer in the same fashion as the Trimble card (Figure 4.3).

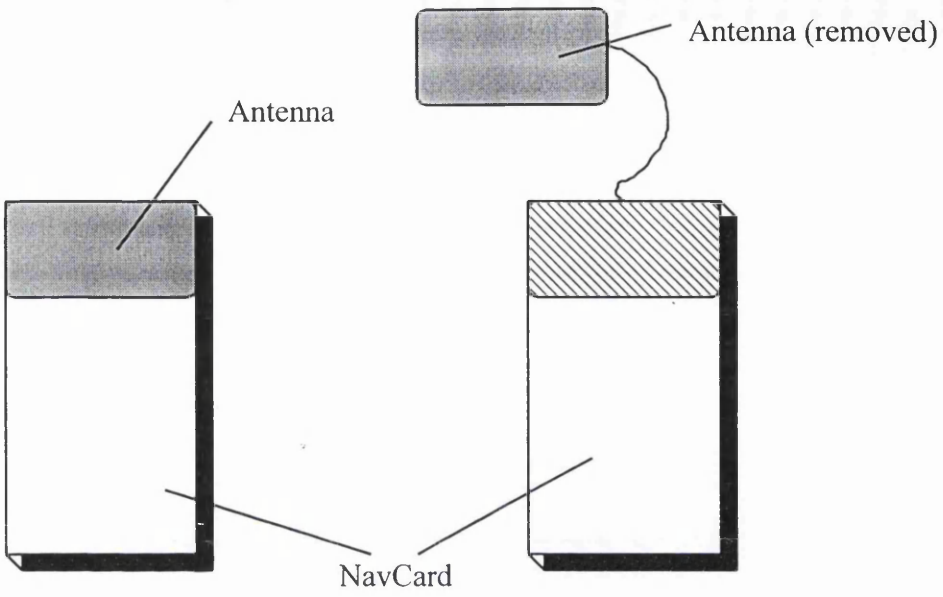


Figure 4.4 NavCard Detachable Antenna

The GPS Silva Compass is a self-contained unit, with extensive navigational software already built in. However, its interfacing with a computer is complex, providing a cumbersome collection of three separate units (the computer, GPS and battery), which could prove difficult to use. The alternative would be to employ the NavCard or Trimble GPS. Both these receivers are personal computer cards which can be directly interfaced with, and powered by, a computer, removing the need for interface cables and external batteries. In terms of performance, both receivers are almost identical. The NavCard, with its detachable antenna provided the most convenient solution; however, the superior software development kit that was provided with the Trimble Mobile GPS meant that it was a better choice for Electronic Navigation Environment investigation and development.

4.3.3 Electronic Compass

No electronic compass was available for design and development due to the unavailability of the relevant components.

4.4 Map Data

There were two main types of data to be considered for use as the digital map display: raster data and vector data (Kennie and Petrie, 1993).

Raster map data is usually derived from a scanned paper map or separates (Owen and Pilben, 1992). Because a raster map is derived from these sources it is the most visually

impressive of the two types, and also most familiar to paper map users. However a raster image consists of a matrix of pixels and is of a fixed size. If the image is manipulated then each individual pixel position must be recalculated. This requires a huge amount of processing power and time. Vector data by its nature is much simpler to manipulate as it consists of points, lines and polygons and so only these elements need to be recalculated, unlike raster data. However, vector maps are less appealing and familiar to the paper map user as they are made up from points and lines, and not actual map images. The scale of the map data depends on the source that the digital map was derived from (Table 4.3).

	Raster	**Vector**
Data Size (20km Tile)	16mb	8mb
Data Scale	1:50 000	1:10 000
Appearance	Familiar	UnFamiliar
Manipulation	Slow	Fast

Table 4.3 Raster and Vector Data
Source: Ordnance Survey (1996)

It was decided that both types of data would be used in the investigation, as by the nature of the navigation environment it should be able to accept the widest range of data possible. Vector map data of The Nottingham Trent University was readily available and so this was selected for use, as well as sample raster data that could be obtained from the Ordnance Survey, Automobile Association, and Bartholemews.

4.5 Summary

In summary, the following software, hardware and map data was selected for use in the investigation into, and development of, the Electronic Navigation Environment. The areas of the project in which the equipment was used are listed in parentheses:

Software:
- MapInfo v3.0 (software design platform, test analysis)
- MapBasic v3.0 (development)
- Visual Basic v4.0 (development)
- Paintbrush v3.0 (icon design)
- PaintShop Pro v3.0 (design diagrams, Software User Interface illustrations)
- Microsoft Draw (design diagrams)
- Trimble Software Development Kit (GPS Driver design, development)

Hardware:
- Viglen DX4-100 Laptop (design, development and testing platform)
- Trimble Mobile GPS Card (development, testing)

Map Data:
- Ordnance Survey 1:1250 Vector Map of City Campus Nottingham Trent University
- Independent survey 1:2500 Vector Map of Clifton Campus Nottingham Trent University
- Ordnance Survey 1:50,000 Raster Map of Port Talbot
- Automobile Association 1:200,000 Raster Map of Norwich
- Bartholomews 1:250,000 Raster Map of Great Britain: Sample

The next stage in the investigation was to design the navigation environment, based on the equipment selected.

CHAPTER 5: ELECTRONIC NAVIGATION ENVIRONMENT DESIGN

5.1 Modules and Module Linking

The Electronic Navigation Environment is a software package that allows the user to perform all navigation tasks in the Microsoft Windows environment. The navigation environment is based and built around Mapinfo GIS, and consists of four separate elements (Figure 5.1).

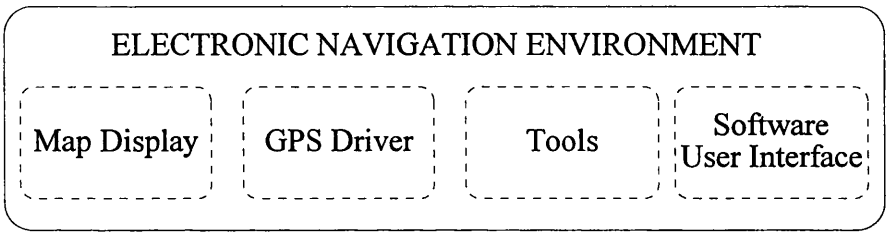


Figure 5.1 Electronic Navigation Environment Components

Each of these elements represents a module which performs specific functions:

Map Display
- Displays digital map data
- Displays positional data

GPS Driver
- Reads data from GPS receiver
- Transforms data to relevant coordinate system
- Plots positional data on digital map, stores data in database

Tools
- Provides the tools and functions to allow the user to manipulate the navigation environment

Software User Interface
- Allows full interaction with Tools and GPS functions
- Enables user to manipulate positional and map data

The Map Display module was essentially Mapinfo GIS modified using MapBasic. The GPS Driver, Software User Interface and Tools were written using a combination of two development languages - Visual Basic and MapBasic. These four modules were linked together in order to create the Electronic Navigation Environment (Figure 5.2).

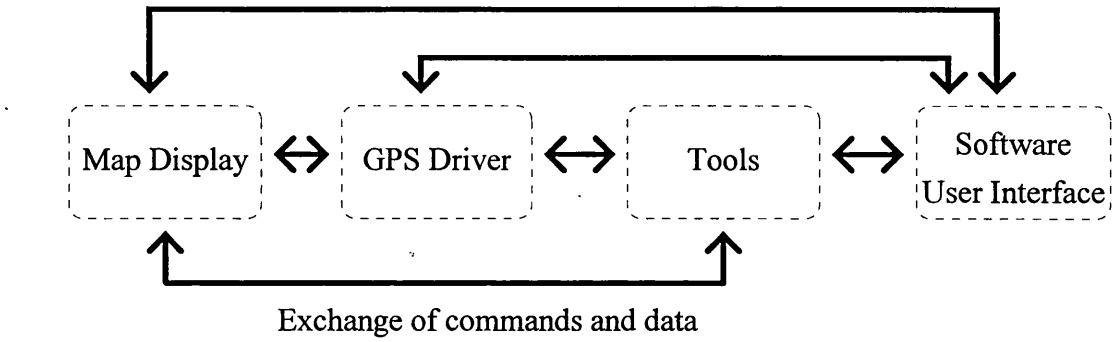


Figure 5.2 Module Linking

5.1.1 Dynamic Data Exchange

In order to link the modules together, Dynamic Data Exchange (DDE) was used. As described by Mapinfo Corporation (1994a) DDE is a means of communications that exists within the Microsoft Windows environment. It allows software that supports the process to exchange data and commands.

A DDE link is created between two or more programs and is known as a conversation. The application that initiates the conversation is known as the client application, with the program being contacted known as the server (Figure 5.3).

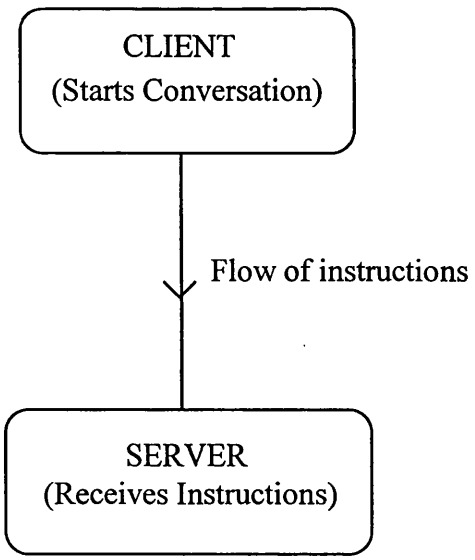


Figure 5.3 Dynamic Data Exchange Link

Once the link has been established then data can be sent from client to server or 'pulled' from server to client. The DDE process is extremely flexible and a program can hold simultaneous conversations with more than one application. A server can also act as a client and initiate conversations of its own even when acting as a server.

The advantages of using such a system are that it relies solely on the Windows environment to operate, allowing real-time exchange of data. The system is also considerably more stable than exchanging data via Disk Operating System (DOS) based ASCII data files, which can be unstable when operating in real time (see Appendix II). The main drawback with DDE is that it can be a complex process to set up and program into the modules. Despite this, it was decided that DDE links would provide the most stable and reliable approach to creating the Electronic Navigation Environment.

5.1.2 Structuring the Electronic Navigation Environment for Dynamic Data Exchange

The following steps were carried out in order to structure the Electronic Navigation Environment for DDE linking:

- Those modules which would communicate with each other were identified, along with the reason for communication.

- The modules were grouped according to programming language.

- Those groups that would act as clients, and those that would act as servers were identified.

Each of the modules with the exception of the Map Display was to be coded using two development languages (see 4.2.2). For simplicity the modules were split into two groups, with respect to development language. Functions performed by the Tools written in MapBasic were combined with the functions of the GPS Driver written in Mapbasic to create a single program group. Similarly those functions of the Software User Interface (SUI) written in Visual Basic were combined with the Visual Basic functions from the GPS Driver (Figure 5.4) .

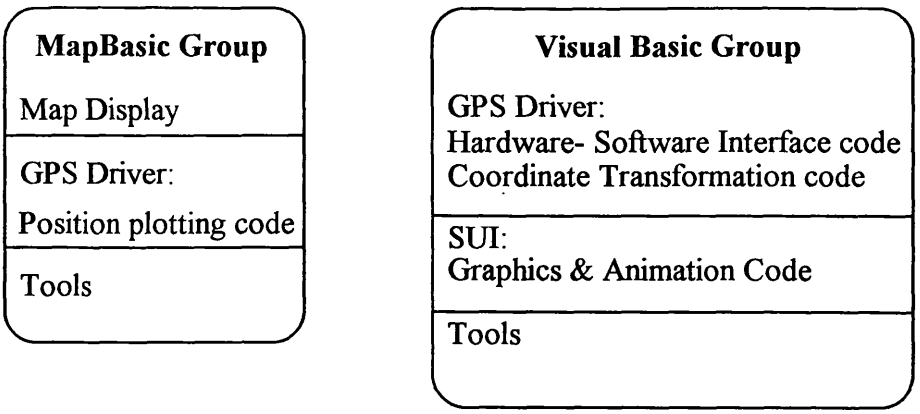
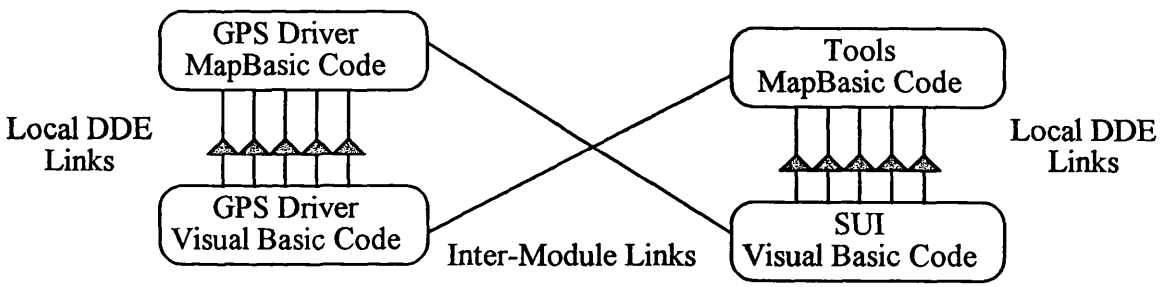
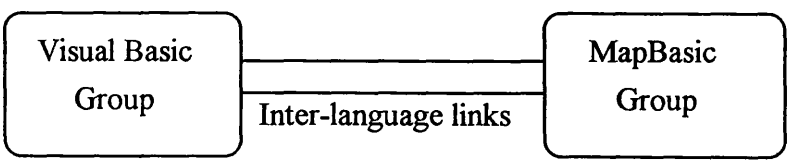


Figure 5.4 Language Groupings of Electronic Navigation Environment Modules

The reasoning behind this process was that any interaction between parts of a module written in different languages would require their own localised DDE conversations. By grouping together the language types, any conversations were centralised (Figure 5.5).



a) Without Language grouping many local DDE links would be required within each module as well as between modules.



b) With language grouping, local DDE converations became unnessecary.

Figure 5.5 Simplification Through Language Grouping

The next stage of the planning process was to determine which parts of each program group needed to communicate with each other. Each menu option and function was viewed as a low-level module and a module relationship chart was established using the guidelines laid down by Brown and Sampson (1973). This chart was derived graphically by dividing the modules into those designed using MapBasic and those designed using Visual Basic. Each

Each module was then linked graphically (Figure 5.6). Once this plan had been created the client and server groups could be established.

The Visual Basic Group was deemed the most suitable application for the client, as it contained the SUI through which the user would interact with the Electronic Navigation Environment. The MapBasic Group would be the server application, reacting to instructions sent from the Visual Basic Group.

As can be seen from Figure 5.6 a large number of functions and interactions existed between the modules, requiring many DDE conversations. It was decided to simplify this process by introducing an indexing system.

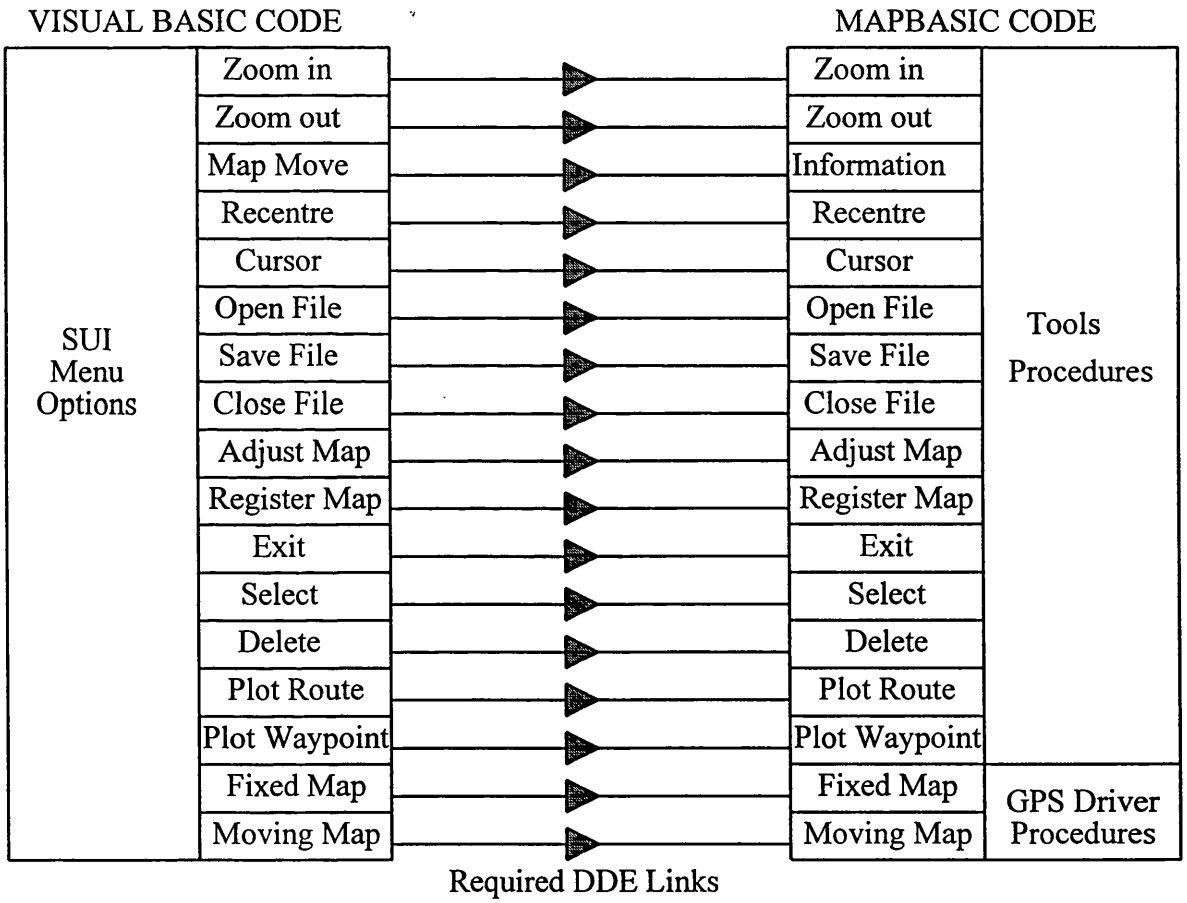


Figure 5.6 Module Linking

5.1.3 Dynamic Data Exchange Index System

The indexing system operated on the principle that each DDE instruction sent by the client (Visual Basic Group) would take the form of an integer. Each integer would refer to a procedure or group of procedures in the MapBasic Program grouping. Thus if the user selects the 'Zoom In' icon on the Software User Interface, an index integer "1" would be sent by DDE to the MapBasic program.

The index integer is received by the MapBasic program group by means of a special procedure known as the Remote Message Handler. As defined by the Mapinfo Corporation (1994b) this procedure is automatically activated when the MapBasic Group receives a DDE executed statement. The Remote Message Handler then enables the program to call the required procedure which, for example, performs the function 'Zoom In' (Figure 5.7). Through using this system, only one DDE link was required, instead of creating a link for each individual function.

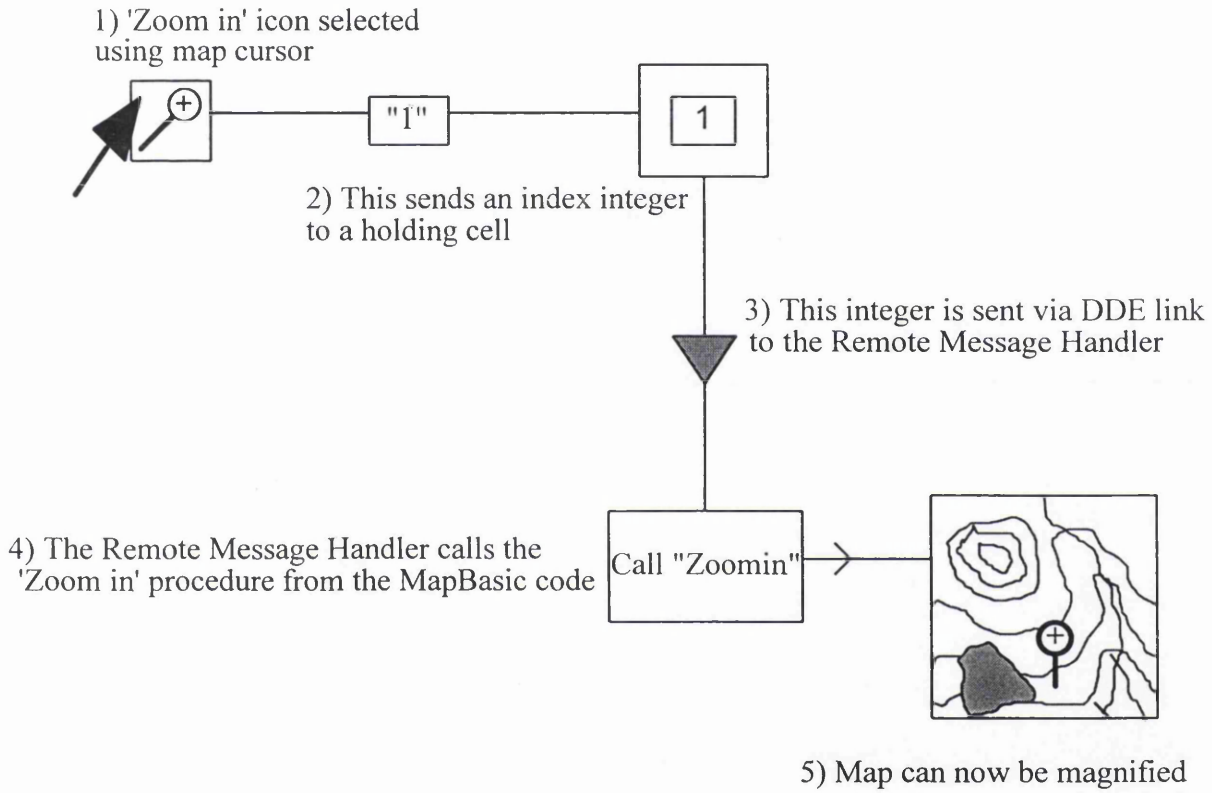


Figure 5.7 Dynamic Data Exchange Index System Dynamics

The DDE process requires all involved applications to be running simultaneously during conversation, so these have to be launched before any conversation can take place. Therefore, the first stage in creating the DDE was to run each program group. Mapinfo is the system on which the Electronic Navigation Environment is based and so this would be activated first. The MapBasic group would then be launched, which in turn activates the Visual Basic group (Figure 5.8). In summary, the overall Electronic Navigation Environment design is as shown in Figure 5.9:

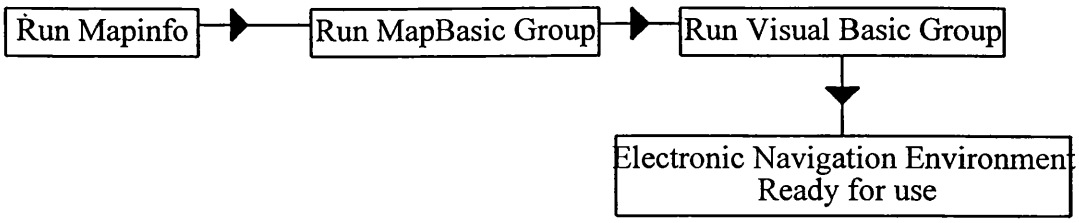


Figure 5.8 Electronic Navigation Environment Operation

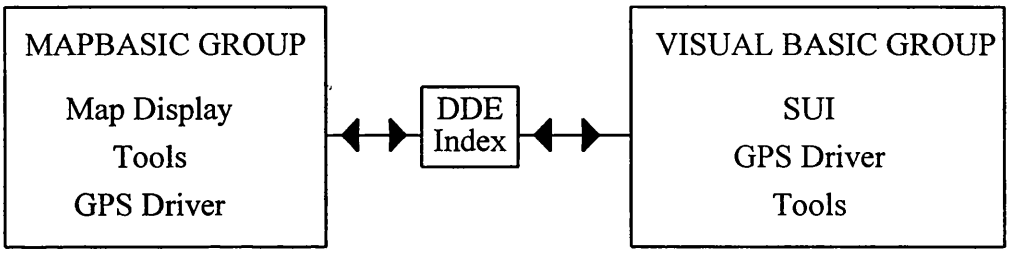


Figure 5.9 Electronic Navigation Environment Design Summary

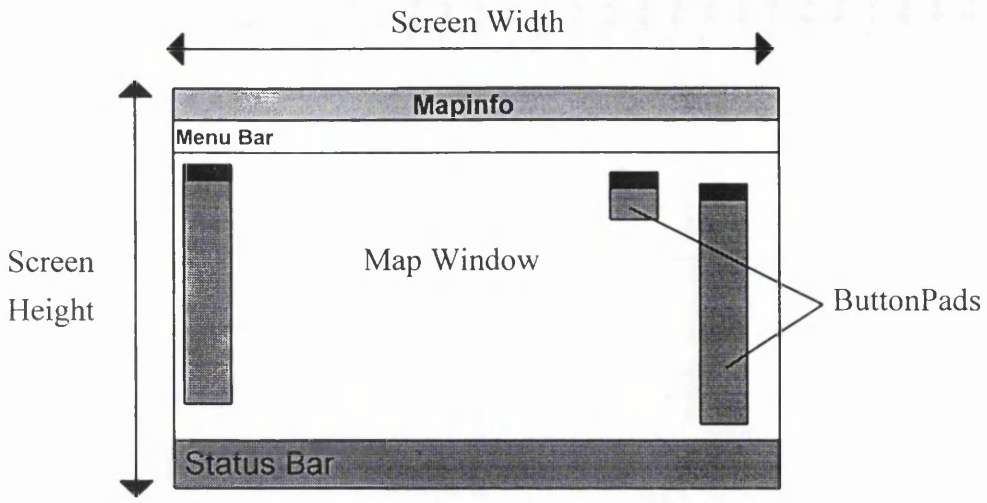
This section established the program design for the Electronic Navigation Environment. The next stage was to design each individual module.

5.2 Map Display Design

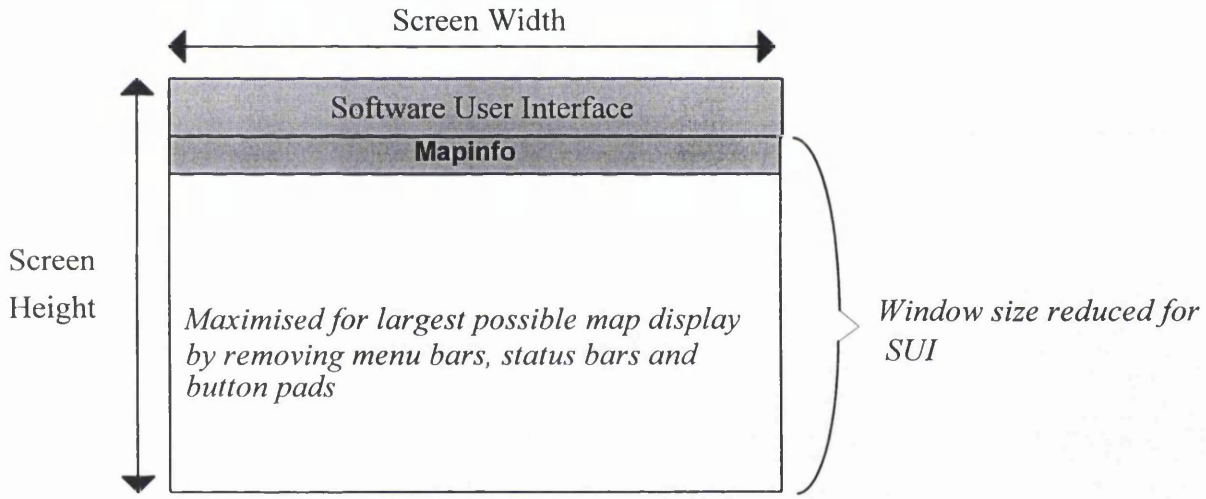
The design of the map display depended on how Mapinfo GIS would be modified to meet the requirements of both the software and the user. The screen layout of MapInfo can be seen in Figure 5.10(a). This had to be altered to maximise the Map Display and remove redundant features.

The first part of the Map Display design was to set the size and position of the map window. The SUI toolbar was located at the top of the screen display and the map display was situated below this, taking up the remainder of the screen (Figure 5.10(b)) In order to achieve this, a procedure was designed to carry out the following tasks:

- Hide the Status Bar
- Position the Window
- Set the height and width of the Window



a) MapInfo Default Display



b) Electronic Navigation Environment Map Display

Figure 5.10 MapInfo to Map Display Modification

These tasks were all carried out using the 'Set Window' command. The programmer input the status bar, position, width and height of the window and this procedure, when executed, will set the window to match these specifications.

Once the window had been correctly positioned the next tasks that had to be carried out were:

- Remove Menu Bar
- Hide Progress Bars

The Menu and Progress Bars were removed to maximise the size of the map display and remove any windows that might have appeared telling the user the status of various operations (Progress Bars). The menu bar was removed as it was redundant, since the Electronic Navigation Environment would use its own menu system. The final stage in setting up the Map Display was to remove the Mapinfo tool bars:

- Remove "Main" Toolbar
- Remove "Drawing" Toolbar
- Remove "Tools" Toolbar

This process was necessary again because the Mapinfo tools were redundant due to the Electronic Navigation Environment SUI and tool set. This process was carried out using the 'Alter Button Pad' command for each toolbar in Mapinfo, with the code located in the main procedure of the program (Figure 5.11).

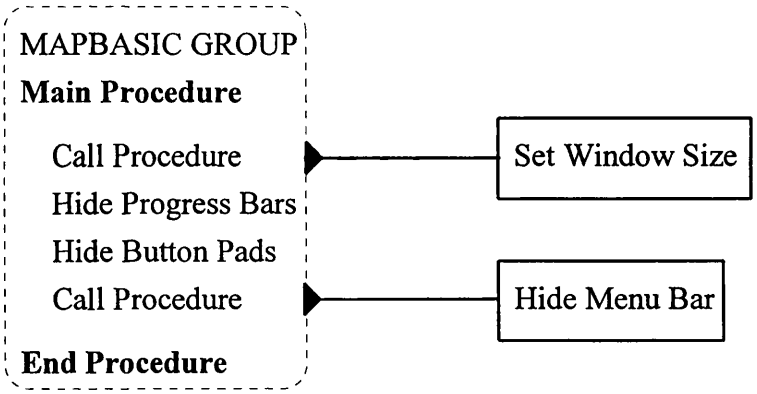


Figure 5.11 Map Display Design Summary

These procedures and functions were designed to be called and executed each time the navigation environment was activated. All the code required for the Map Display was located in the MapBasic Group.

5.3 Tool Design

This section covers the design of the tools and functions available in the Electronic Navigation Environment. The tools and functions were divided into the following sections:

- Map Manipulation
- Route Planning
- Data Storage and Retrieval

The Tools were designed as independent modules using the methods discussed by Brown and Sampson (1973).

5.3.1 Map Manipulation Tools

Zoom In
Zoom Out
Move Map
Recentre

Zoom In allows the user to magnify the map, in order to view parts of the map display in greater detail. **Zoom Out** enables the user to reduce the magnification of the map. Both of these tools are provided for development with MapBasic in the form of menu commands, which can be called when the relevant tool is invoked.

Move Map enables the user to pan around the map. Such a function is used by Mapinfo in the form of a 'Grabber' (whereby the user drags the map across the screen to the desired location using a mouse or similar device). However, this function was not available for development, therefore the tool was designed as scroll bars on the map window, to enable the user to pan and move the map. The **Move Map** procedure had to:

- Select Map Display Window
- Add scroll bars

Recentre enables the user to centre the map display on the last recorded position. This function required the coordinate values of the last known position, which were then used to centre the map. The map display would then be redrawn accordingly:

- Recall last recorded Easting, Northing Coordinate Values from Database
- Create Point Using Easting, Northing
- Map Centre = Easting, Northing
- Map is then automatically redrawn with Easting, Northing at its centre

5.3.2 Route Planning Tools

Plot Waypoint
Plot Route
Measure
Select
Delete
Cursor
Time Estimation

Plot Waypoint allows the user to plot waypoints on the map. A MapBasic command allows the plotting of points, but the point style must be designed before the command is invoked. The point must stand out on any type of map data, whether colour or monochrome. The following design was chosen (Figure 5.12).

Figure 5.12 Waypoint Design

The Waypoint marker was designed as a circular symbol, coloured bright yellow with the annotation 'WP' in the centre. A circle was used as the shape to signify that the symbol represented a single point. A line symbol such as a cross could have been used but these stand out poorly when plotted on a map. The bright yellow colour was used as according to Sutcliffe (1995) yellow has good visibility and visual acuity which attracts the user's attention to the point. A thick black surround was used to add emphasis to the symbol. Finally, the WP annotation signified that the symbol represented Waypoint. Therefore the **Plot Waypoint** procedure had to carry out the following tasks:

- Set style of the waypoint symbol
- Plot point at user's discretion

The **Route Plotting** tool allows the user to plot their intended route on the map. The **Route Plotting** tool must have its line style already set when this tool is selected. Magenta was used on a solid line (Figure 5.13).

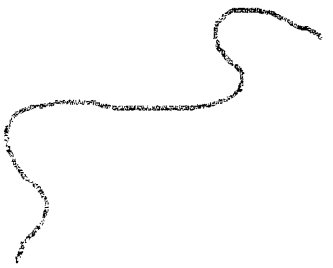


Figure 5.13 Route Line Style

Magenta was used as it naturally attracts the human eye and draws attention. It is also little used on most digital maps available for the UK and as noted by Travis (1991) is easily distinguished from the main colours used on UK digital maps. According to Lawrence (1971) these are: black, red, green, yellow, blue and brown. A solid line was used for emphasis as dashed or dotted lines lose importance. This tool involves:

- Set style of line symbol
- Plot route at user's discretion

The **Measure** function enables the user to measure straight line distances between points or a series of points. A window displays the total length of the line(s) measured as well as the length of the last leg if a polyline has been measured. This function was provided as a MapBasic command.

The **Select**, **Delete** and **Cursor** commands allow the user to edit information plotted as an overlay on the map display, such as Waypoints or Route lines. **Select** allows the user to draw a box around all the objects they wish to select. If only a single object is to be edited, then the user can select this object using the **Cursor**. Finally, the **Delete** tool removes any selected objects from the map display (N.B. The background map is frozen so items cannot be deleted from it). These tools were provided as MapBasic commands.

The **Time Estimation** tool enables the user to calculate journey times based on the length of the journey in terms of distance and their average walking speed. This involves the user entering a speed and distance, and the tool calculating the resultant journey time. Distance has to be entered to the nearest kilometre, with speed to the nearest kilometre per hour. Once these values are input the tool calculates time using:

Time = Distance / Speed

The resultant time is then converted from a decimal to a sexigesimal value and output to the user via the SUI. This procedure was coded in Visual Basic, and involved the following steps:

- Input Speed
- Input Distance
- Calculate Time = distance / speed
- Convert Time to Hours/Minutes
- Output Time

5.3.3 Data Storage and Retrieval Tools

Open File
Close File
Save File
Map Set Up
Adjust Map
Exit

Open File, **Close File** and **Save File** allow the user to carry out data storage and retrieval tasks that are necessary with any computer system. **Open File** allows the user to load new or existing map data, as well as load data files such as existing route plans. **Close File** allows the user to remove any opened file from the system, without deleting it. Finally **Save File** enables the user to save any changes they have made to displayed data, as well as saving positional and route information. These functions were all available as MapBasic commands.

Map Set Up is required if a new raster base map is to be used within the navigation environment. According to Mapinfo Corporation (1994a) a new map must be registered, so that coordinate values can be related to it, as raster map data is a 'dumb' matrix of pixels to which coordinate and projection information cannot be directly attached. Vector data is stored as a series of coordinate values, and so projection and coordinate information can easily be attached, thus the **Map Set Up** procedure is not required for vector data. For raster data, the map set up process involves four steps:

(1) Select Projection System: The user must state which projection system the map they are using is based on. The menu will offer a selection of projection systems which can be highlighted if required.

(2) Select Coordinate System: The next step is to select the appropriate coordinate system to be used with the base map. This can either be a grid based system which will display position in X and Y (or Eastings and Northings with the OS National Grid), or a system based on a graticule which displays position in latitude and longitude.

(3) Select Map Units: This specifies which units any measurements made on the base map will be displayed in (e.g. metres, kilometres or degrees).

(4) Register Map: The final step is to register the map. This is achieved by selecting four or more points on the base map for which the user has known coordinate values. This step references the base map to the coordinate and projection system selected in steps (1) and (2).

It is vital that this procedure is carried out, otherwise GPS data will not be plotted in the correct position on the base map.

The **Adjust Map** function enables the user to alter the contrast and brightness of the base map in order to achieve the sharpest image possible. The Adjust Map and Map Set Up functions were provided as MapBasic commands.

The **Exit** command allows the user to terminate the Electronic Navigation Environment and revert back to the standard Mapinfo GIS system. This involves the termination of all DDE links as well as a statement to end the Electronic Navigation Environment program:

- End all DDE Conversations
- End Program

5.3.4 Tools Design Summary

The Tools were designed to be controlled by the user via the SUI. The procedures discussed would wait until the relevant command was received from the SUI before executing (Figure 5.14). Those tools written in MapBasic would be activated via the DDE link, using the indexing system discussed in section 5.1.

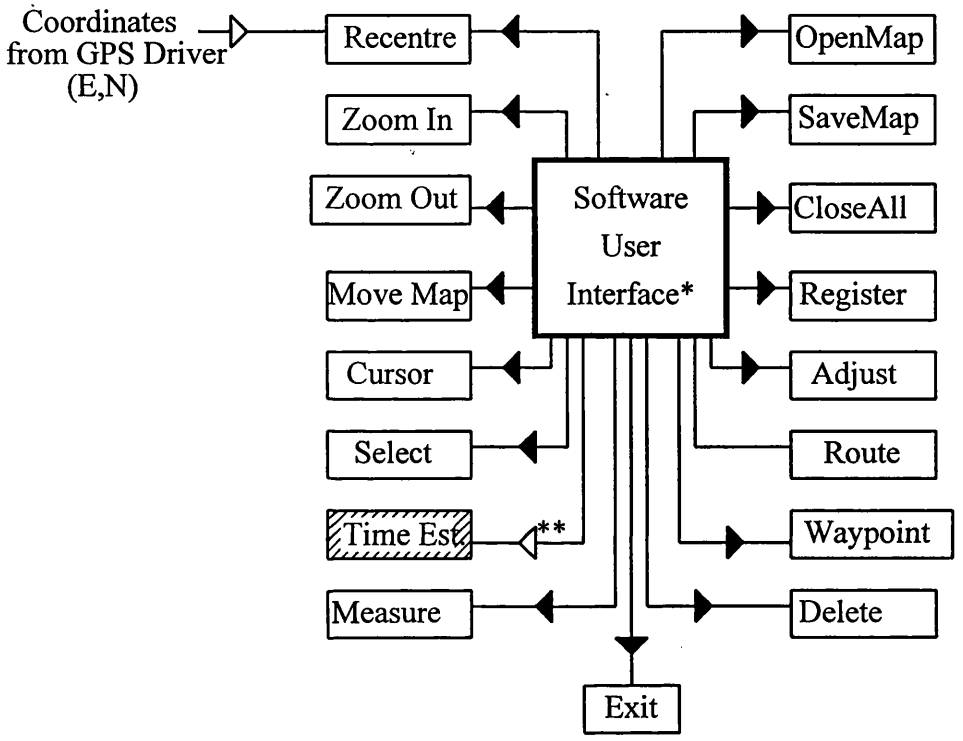


Figure 5.14 Tool Program Plan

5.4 GPS Driver

The GPS Driver was designed using a combination of methods. A Top-Down approach, as laid down by Cassel (1983), was used to establish the basic structure of the GPS Driver. A more detailed design of the processes required within this structure was determined using the Jackson Design Methodology as defined by King (1988).
A GPS Driver is a software application that links the hardware of the receiver to a digital map display and / or data file (Figure 5.15).

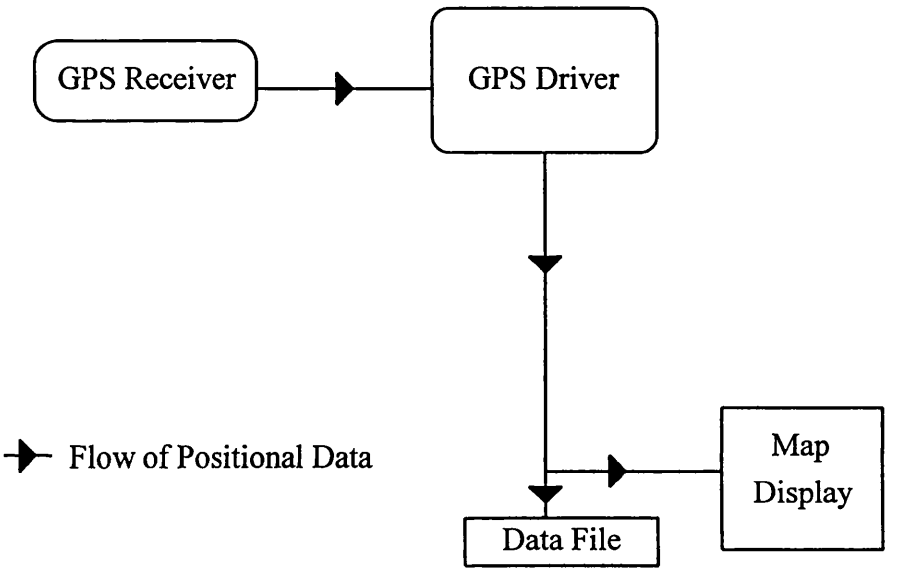


Figure 5.15 GPS Driver

In terms of Electronic Navigation Environment design, the GPS Driver had to successfully link the Trimble Mobile GPS to the Map Display, so that GPS position could be plotted in real time, as well as being automatically stored in a database. With this in mind, the GPS Driver was divided into two distinct blocks (Figure 5.16).

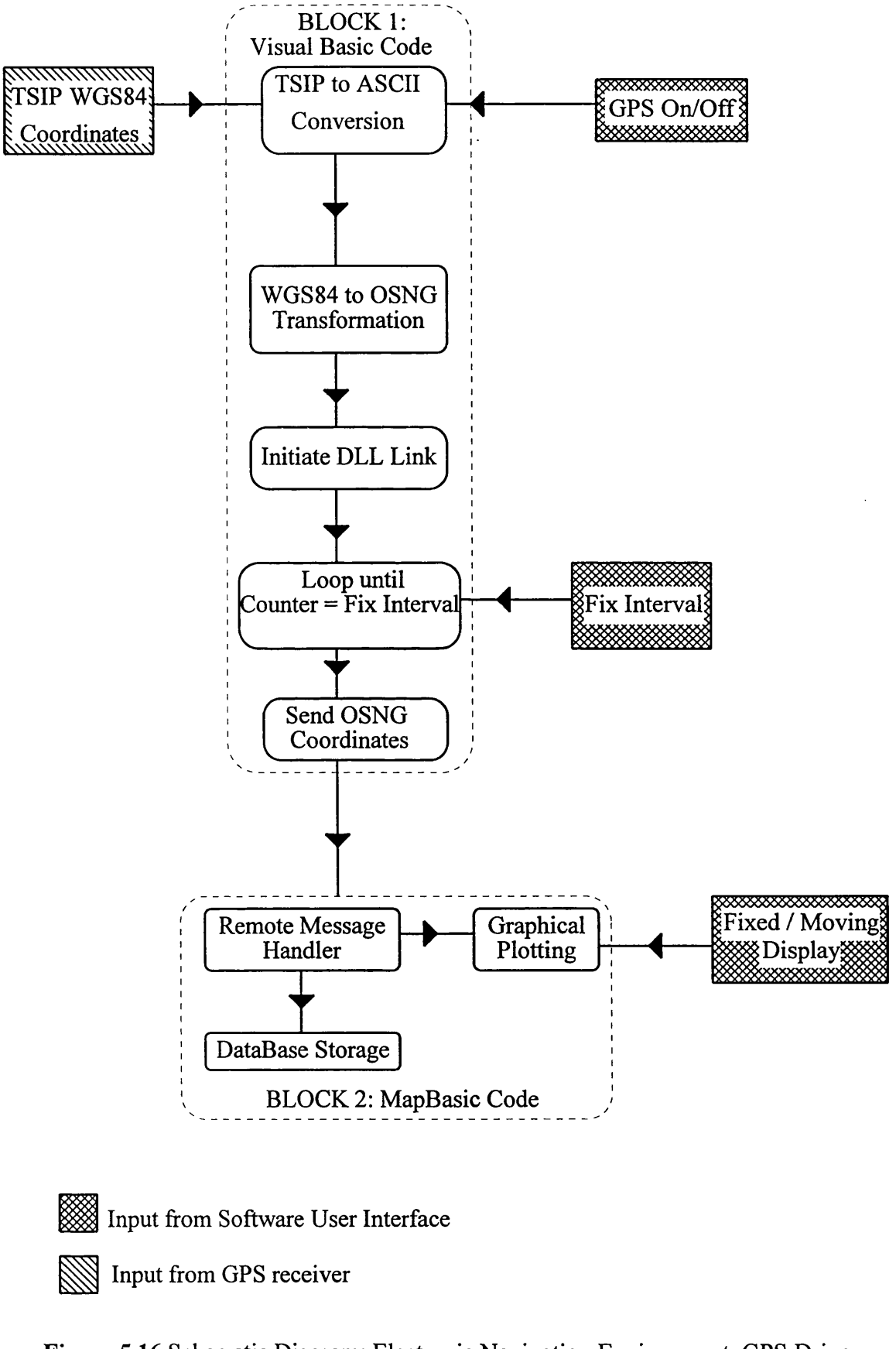


Figure 5.16 Schematic Diagram: Electronic Navigation Environment GPS Driver

The first part of the GPS Driver (Block 1) had to collect the data from the GPS receiver and convert it to a format readable by the Map Display. This would then send the readable data to the second part of the driver (Block 2) which was to consist of data plotting and storage functions.

5.4.1 Block 1: Data Collection and Transformation

The first stage in designing the data collection and transformation block was to investigate how the receiver would interface with any software, and the nature of the data being supplied by the GPS receiver.
As stated by Trimble Navigation (1994a) the GPS receiver communicates with all software applications via a Dynamic Link Library (DLL). A DLL as defined by Swann (1995) is a library of commands and structures that can be accessed by any Windows program during run time.

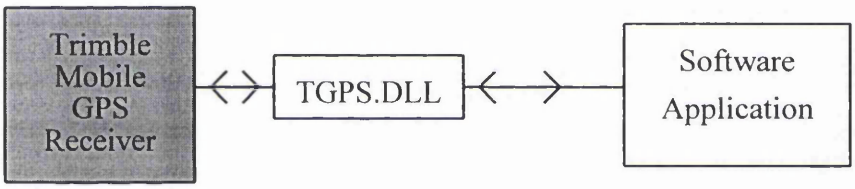


Figure 5.17 Dynamic Link Library (DLL)

As can be seen from Figure 5.17 the TGPS.DLL allows a software application to communicate with the Trimble Mobile GPS receiver. This includes commands to initiate a link between receiver and software as well as commands to receive data. The Trimble GPS Control Panel must be launched before any other applications, and so this must be a part of the GPS Driver design.

As described by Trimble Navigation (1994b) the data is supplied in a binary format known as TSIP (Trimble Standard Interface Protocol) in a series of packets, which can range from co-ordinate information to the general 'health' of the receiver. The receiver can be altered to communicate in two other protocols - TAIP (Trimble ASCII Interface Protocol) and NMEA, a binary format designed as a marine industry standard. The use of these alternative protocols would involve reconfiguring the embedded software of the receiver which is potentially a very complex process, so it was decided to use the default setting as this would be the system most likely used by the land navigator. The TSIP format could not be read by the Map Display and had to be converted to ASCII format.

The main data required from the GPS Driver is co-ordinate information. This is supplied by the receiver in the WGS84 co-ordinate system. In order for the navigation environment to operate successfully with digital maps of Great Britain, co-ordinates are required in the Ordnance Survey National Grid (OSNG) system. Therefore, a co-ordinate transformation was required. Finally, the converted and transformed co-ordinates were to be sent to the data storage and plotting block of the GPS Driver. To summarise, Block 1 was to involve:

- Initiate receiver - software link (using Dynamic Link Library)
- Collection of WGS84 co-ordinates in TSIP Binary format
- Conversion of WGS84 co-ordinates in TSIP format to WGS84 co-ordinates in ASCII format
- Transformation of WGS84 co-ordinates to OSNG
- Send OSNG ASCII co-ordinates Block 2.

5.4.1.1 Initialisation, Collection and Conversion

The Initialisation of the receiver - software link was to take place utilising the commands defined in the DLL. This would also launch the GPS control panel, which was to be hidden from the user's view unless selected via the SUI.
The Collection and Conversion of the Data from the GPS receiver was to be carried out using the functions and commands supplied in the DLL. The Visual Basic Data logging program example supplied with the Software Development Kit (SDK) was modified to print the co-ordinate values into a window as they were converted. The user would be able to initiate or terminate the collection and conversion process via the SUI by either turning the GPS receiver On or Off.

5.4.1.2 Co-ordinate Transformation

Two methods were considered for the co-ordinate transformation process. The initial stage in selecting the most appropriate method was to define the requirements of such a transformation.
The transformation had to be two-dimensional, as a height value was not required for plotting on the map display. The transformation had to be able to perform in real time (as positions had to be plotted on the map in real-time), providing a solution for a single point. Therefore post-processing methods were unsuitable.
The accuracy requirements of the transformation were fairly complicated, as any error introduced at this stage would be adding to errors already present in the co-ordinate values

caused by signal downgrading. Raw co-ordinate data entering the program is accurate to within +/- 100 metres (as explained in 1.2.5) and so a transformation accuracy of +/- 10 metres was chosen as an initial specification, at most a 10% increase in the error of the co-ordinate values. This seemed to be more than acceptable when considering the accuracy of the raw data and the accuracy of the map data on which the co-ordinates would be plotted (distortions and displacements of up to 50 metres at 1:50 000 scale). It was also important to achieve consistency throughout Great Britain. To summarise, the specifications for the transformation were:

- Real-time Capability
- Two-dimensional Solution
- Accuracy of +/- 10 metres
- Consistent accuracy throughout Great Britain

With these requirements in mind, the Molodensky and Ordnance Survey transformation methods were considered for use in the GPS Driver. A full explanation of these methods is given in Appendix III.

The suitability of each method depended on how it compared to the transformation requirements. The Molodensky transformation was capable of providing a two dimensional solution with the non-inclusion of height in the process greatly simplifying the procedure. Despite this, the transformation failed to meet the requirements in terms of accuracy and consistency throughout Great Britain. This was due to the shift parameters used when transforming from WGS84 Cartesian co-ordinates to Airy Cartesian co-ordinates. According to the Defense Mapping Agency (1991) these shifts are only accurate to +/- 20 metres and vary depending on location in Great Britain by up to 14 metres (Table 5.1). This method did not take into account errors that exist in the OSNG, which could further reduce the accuracy of the final co-ordinates in relation to the user's position on the digital map.

Region	Transformation Parameters		
	X	Y	Z
Mean Solution	375 (+/-10)	-111 (+/-10)	431(+/-15)
England	371 (+/-5)	-112 (+/-5)	434 (+/-6)
Scotland	384 (+/-10)	-111 (+/-10)	425 (+/-10)
Wales	370 (+/-20)	-108 (+/-20)	434 (+/-20)

(N.B. Numbers in brackets show error estimates of datum shift parameters. These estimates DO NOT take into account any errors that exist in the control point co-ordinates and so the actual error values could be much larger).

Table 5.1 Variation in Transformation Parameters used in Molodensky Transformation WGS84 to OSGB36

Source: Defense Mapping Agency (1991)

The Ordnance Survey method was designed specifically to provide a two-dimensional solution. The transformation was quoted by the Ordnance Survey (1995a) to be "at the 2 metre level of accuracy", which was well within the specified accuracy. Its consistency throughout Great Britain was also excellent as the method had been designed with any errors in the National Grid in mind (taken into account in the shift parameters). These factors meant that this method was more suitable for use in the transformation from WGS84 co-ordinates to OSNG co-ordinates.

5.4.1.3 Send Ordnance Survey National Grid Co-ordinates to Data Storage and Plotting

This process involved sending co-ordinates from Block 1 to Block 2. In order to achieve this DDE was to be used. This involved establishing a communications link to send the co-ordinates from Block 1 to Block 2. The frequency at which this procedure would be executed depended on the Fix Interval (i.e. the time interval between GPS position fixes). This was to be entered by the user via the SUI and activate a loop, which would only send co-ordinates to be plotted after the appropriate interval had elapsed (Figure 5.18).

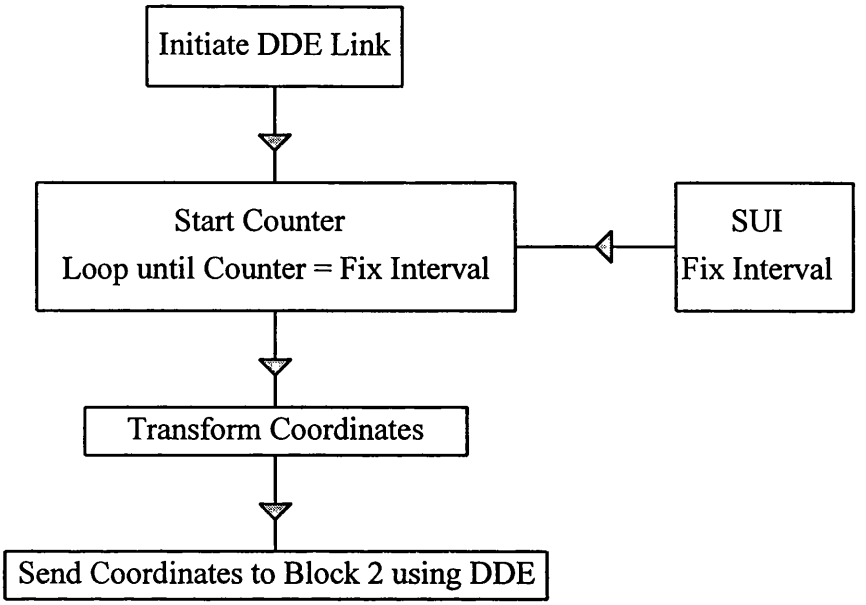


Figure 5.18 Fix Interval Loop

5.4.2 Block 2: Data Storage and Plotting

The Data Storage and Plotting Block was designed to perform the following functions:

- Receive ASCII OSNG co-ordinate data via DDE Link.
- Send co-ordinates to database for storage
- Plot co-ordinates graphically on Map Display

This series of functions was to be carried out using the following MapBasic procedures. The co-ordinates were to be received from the DDE link by the Block 2 Remote Message Handler, which would operate by lying dormant until it received data via the DDE link. The procedure would then activate and perform the functions specified within it. In this case, the Remote Message Handler was to assign the co-ordinates received to variables, and then call a procedure that would store the co-ordinates in a database then plot the co-ordinates on the map display (Figure 5.19).

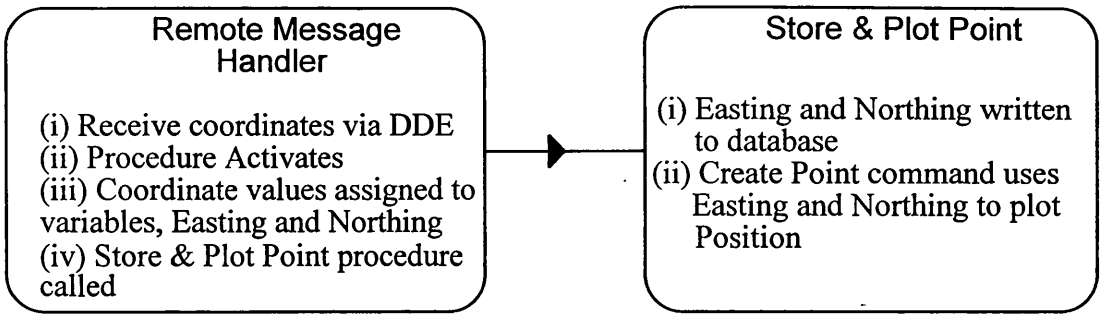


Figure 5.19 The Remote Message Handler

(i) Database Storage

The values received from the Remote Message Handler were to be immediately written to a database. The database simply consists of a list of Easting and Northing fields to which co-ordinates would be written:

- Get Co-ordinate Values
- Insert Values into DataBase.

(ii) Graphical Plotting

Plotting positional information on the Map Display could be achieved in two ways:

- Fixed Map, Moving Position
- Moving Map, Fixed Position

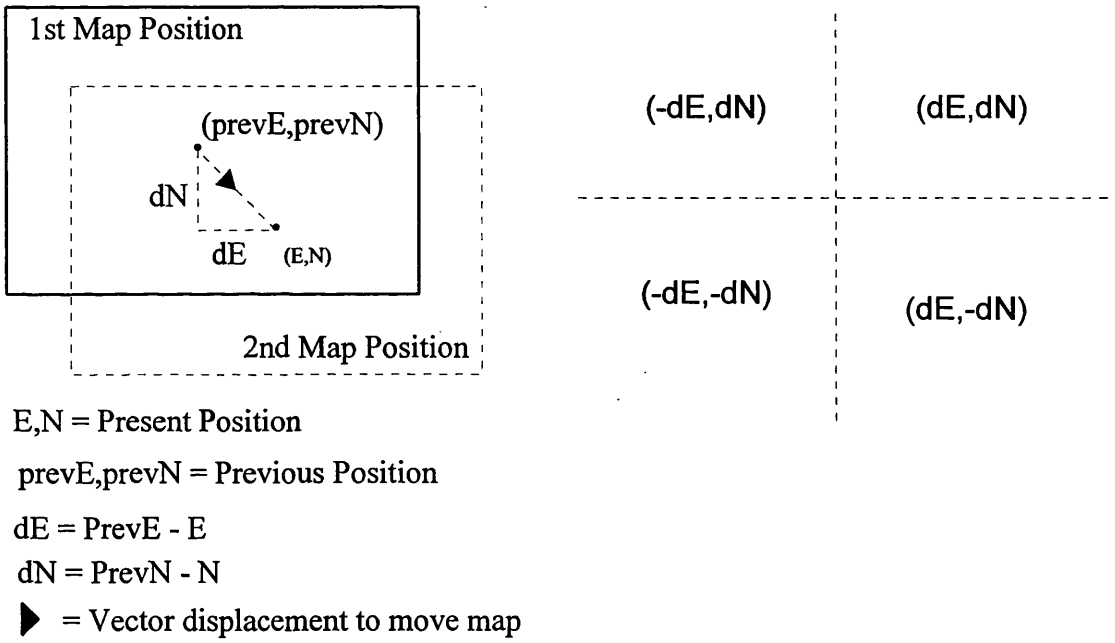
The fixed map solution means that as the user's position is plotted on the map, the map remains stationary whilst the position moves on the screen. This would simply involve plotting the point directly onto the map using the co-ordinates provided. The fixed map solution would select the appropriate co-ordinates from the database and then plot these points on the map:

- Fetch Co-ordinate values from database
- Plot point using selected co-ordinates

The moving map solution was more complex and so required a more complex procedure to operate:

- Fetch Co-ordinate values from database
- Establish magnitude and direction of movement
- Plot point using selected co-ordinates

The magnitude and direction of movement could be determined using the differences between the previous position and present position. Figure 5.20 shows how the difference in co-ordinate values could determine the direction of movement. For example, if the difference between prevN and N was negative, this constituted a southerly movement of the map, as the position would be advancing north.



E,N = Present Position

prevE,prevN = Previous Position

dE = PrevE - E

dN = PrevN - N

▶ = Vector displacement to move map

Figure 5.20 Determination of Map Movement

These two solutions for displaying the users position were to be provided by the GPS Driver, with the user selecting their preference via the Software User Interface.

(iii) Position Cursor

In order to plot position using the methods outlined above a position cursor was required. The design was limited by the array of symbols available within Mapinfo. The symbols present could be edited and customised, but only to a limited extent. The following design was chosen (Figure 5.21).

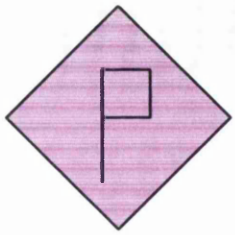


Figure 5.21 Position Cursor

The cursor chosen was diamond shaped, with a black outline, magenta infill and annotation 'P'. The diamond shape was chosen as it is a simple geometric shape that is easily discrimintated by the visual system (Keates, 1989). It is also much easier for the user to visualise the centre of such a shape than a square or rectangle. The magenta colour was used to give the symbol strong visual impact and good contrast, in the same manner as the plot route tool (Travis, 1991). The black outline was used to emphasise and highlight the cursor's shape using the guidelines laid down by Horton (1994). Finally it was annotated with the letter 'P' to denote position and provide clarity to the cursor's purpose.

5.4.3 GPS Driver Summary

In summary, the GPS Driver was designed to perform the following tasks (Figure 5.15):

Block 1
- Collection of WGS84 co-ordinates in TSIP Binary format
- Conversion of WGS84 co-ordinates in TSIP format to WGS84 co-ordinates in ASCII format
- Transformation of WGS84 co-ordinates to OSNG using Ordnance Survey Method.
- Initiate DLL Conversation
- Send ASCII OSNG co-ordinate data

Block 2
- Receive ASCII OSNG co-ordinate data via DLL Link
- Send co-ordinates to database for storage
- Retrieve co-ordinates from database and plot graphically on Map Display

5.5 Software User Interface Design

The final design process was the design of a suitable Software User Interface. A computer program operates by the user issuing a set of commands, and the computer carrying these out. The first step in designing the user interface was to name the commands necessary to allow the user to carry out the functions and tools designed earlier in the chapter, in accordance with the methods outlined by Sutcliffe (1995).

5.5.1 Command Naming

GPS - The GPS had to be able to be switched on and off, and have its fix interval determined, and the display type selected. The Trimble GPS Panel had to be selectable: **GPS On, GPS Off, Fix Interval, Fixed Map Display, Trail, Moving Map Display Advanced Options** (Trimble GPS Panel).

Compass - The Compass had to be able to be displayed or hidden (so as not to obscure the map display permanently), therefore two commands were required: **Display Compass, Hide Compass.**

Map Manipulation Functions - The functions required for map manipulation included magnify map, reduce map, move map, recentre map and a cursor to allow the selection of the various tools and parts of the map display: **Zoom In, Zoom Out, Move Map, Recentre, Cursor.**

Measuring, Marking Functions - The map must be measured and have routes and points plotted on it or deleted from it. The user had to be able to estimate journey times, which required the input of speed and distance, and the output of time: **Plot Waypoint, Plot Route, Delete, Select, Measure, Speed, Distance, Time Estimation.**

Data Storage and Retrieval Functions - These functions would allow the user to retrieve and store different maps, routes etc: **Open File, Close File, Save File.**

Map Set up and Adjustment Functions - These functions must allow the user to register a new map, or adjust one that has been improperly set up: **Map Set Up, Adjust Map.**

Help Functions - These functions were to provide the user with aid if they came into difficulty with the software: **Help.**

One other command was essential to the program, which is: **Exit.**

These were the basic commands that had to be made available to the user, in order for them to interact effectively with the Electronic Navigation Environment modules. Next, these commands were ordered and structured in a menu system.

5.5.2 Menu System

The first stage in creating the menu was to establish a function hierarchy. This was based on how often functions would be used, and how important the functions would be to the overall program (Figure 5.23).

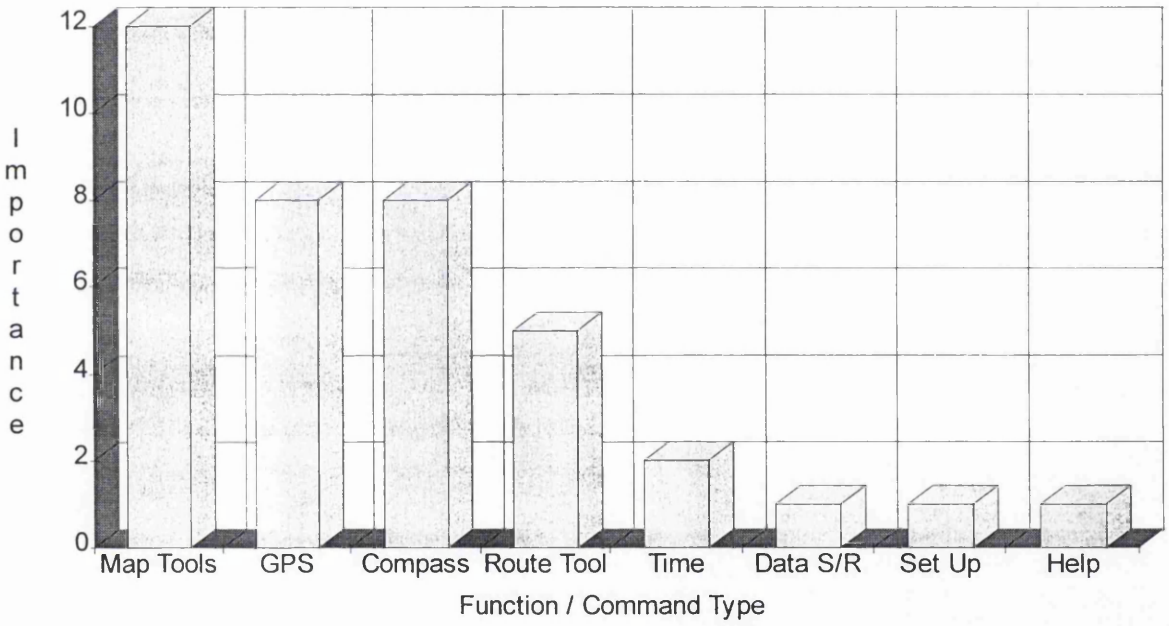
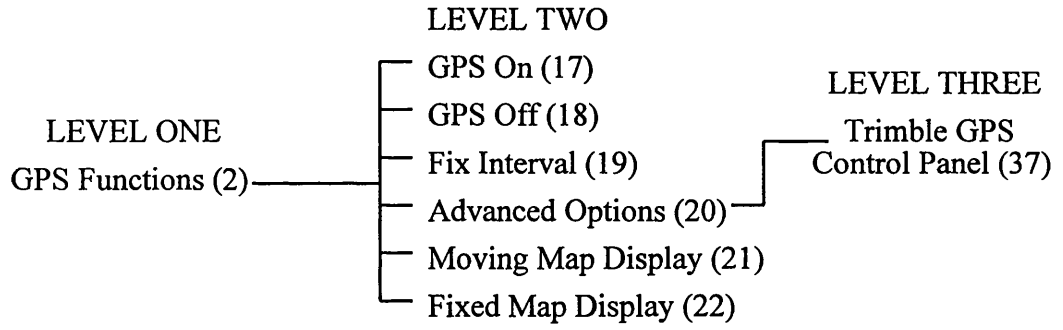


Figure 5.23 Command and Function Hierarchy

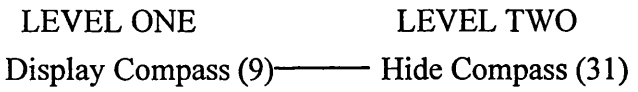
The hierarchy created in Figure 5.23 was determined by noting the frequency of the functions in the Navigational Tasks of the User Profile, as discussed by Shneiderman (1987). Each function was weighted, depending on the navigational task it was required for and how important this task was (see Appendix IV).
As can be seen from Figure 5.23, the most important functions are the map tools so these were designed to appear at level one in the menu i.e.:

LEVEL ONE:
Zoom In (4)
Zoom Out (5)
Move Map (6)
Recentre (7)
Cursor (8)
(N.B. Numbers in parentheses refer to Figure 5.24)

GPS and Compass commands followed in order of importance and were assigned to level two. They would be accessed via Level one, but no GPS commands were situated at level one.

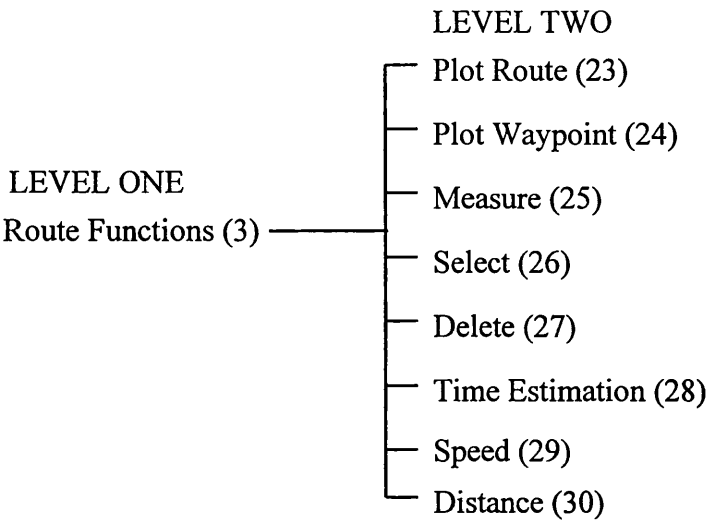


Advanced Options were located at level three as they would be used infrequently.

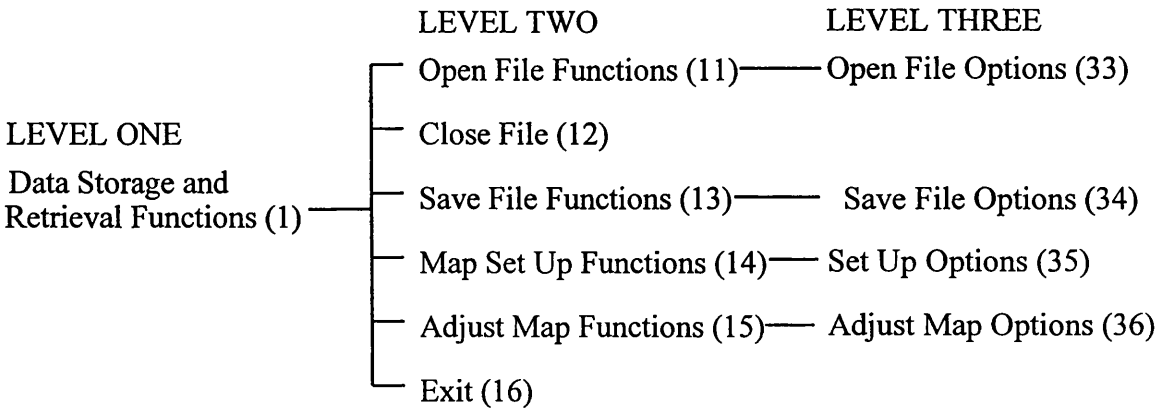


The compass display command was situated at Level One, which displays the compass graphic. Once the graphic has been displayed, it reveals the Hide Compass command at Level Two.

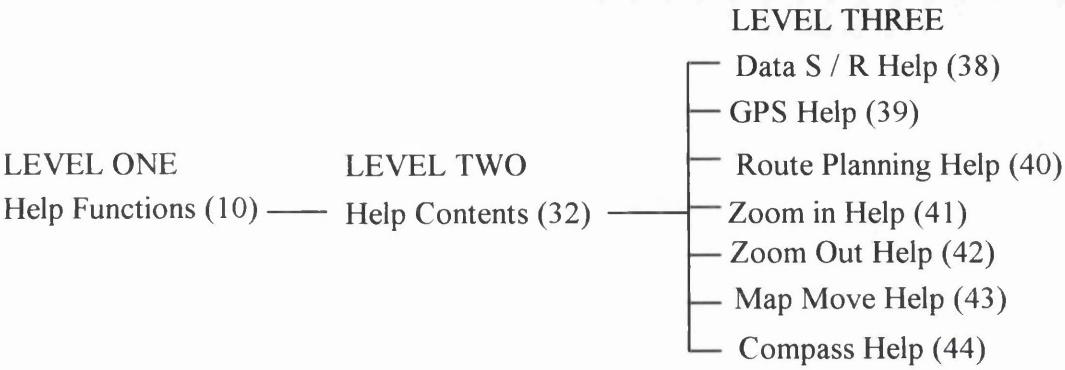
The next set of commands in the hierarchy were the Route Tools and Time Estimation tools. These commands were combined, as they were all associated with measuring, marking and estimating routes. They were situated at Level Two, with access via Level One.



Finally, there are the Data Storage and Retrieval, Map Set up and Help Functions. These functions would all be used infrequently and are not vitally important to navigation. However, they are required by the software for the system to operate effectively. These Functions were situated at Level Three, with access through Levels One and Two. Data Storage and Retrieval and Map Set Up functions were grouped together and accessed via a single point at Level One, followed by a sub menu at Level Two:



This same organisation was applied to the Help Functions:



Once the hierarchy and grouping of the functions was determined, they were added together to create the overall menu system (Figure 5.24). Thus the menu breadth was established as ten and depth at three.

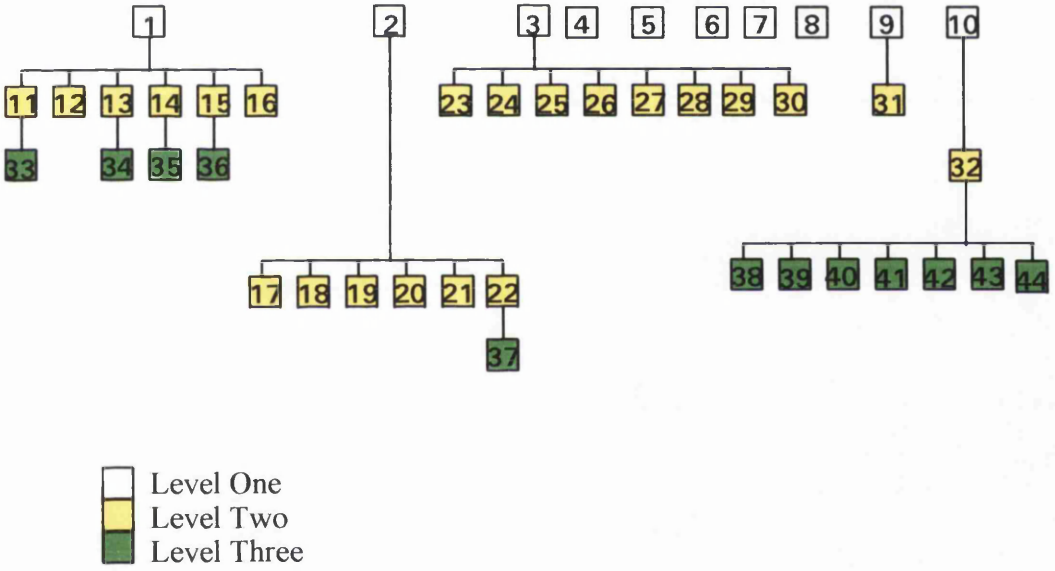


Figure 5.24 Electronic Navigation Environment Menu System

The next stage involved deciding how each level and command was to be represented.

5.5.3 Representation of Options

A Graphical User Interface (GUI) was designed instead of a Command Language or Natural Language based interface. The GUI method of representation was chosen for the following reasons:

• Due to the physical environment in which a personal navigation device will be used, a

keyboard is not very practical to use as the main input device. A GUI can rely solely on a pointing device for input, removing the need for a keyboard. Command and Natural Languages require a keyboard (Helander, 1992).

- Discretionary usage is voluntary (for explanation see 3.2.2) so the system needs to be easy and quick to learn. This rules out a command language. A well designed GUI should be quick and easy to learn.

- Frequency of use is low and so once again the interface should be quick and easy to learn.

This representation of options was organised into four parts:

(i) Screen Layout
(ii) Level One Representation
(iii) Level Two Representation
(iv) Level Three Representation

(i) Screen Layout

The map is the most important function to be displayed by the Electronic Navigation Environment, and so logically this was to be the most dominant feature. The ten Level One commands and functions were to be displayed in a Toolbar at the top of the display, with Level Two and Three options hidden in pop-up menus that could be viewed and selected when required. (Figure 5.25).

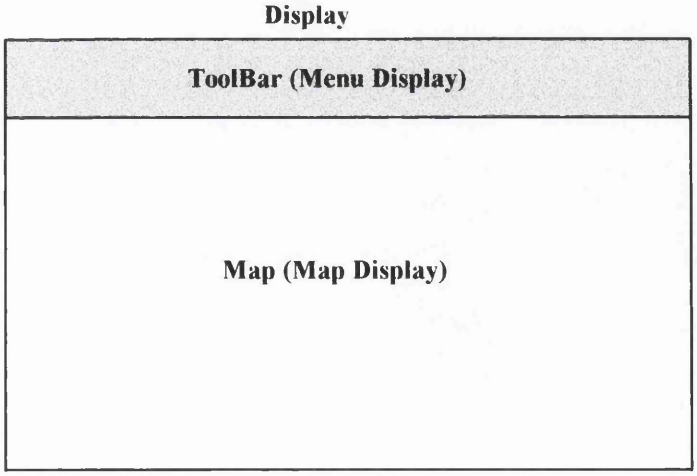


Figure 5.25 Screen Layout

(ii) Level One Representation

All Level One options were to be presented to the user in iconic form. Each of these icons was carefully designed in order to provide the options in a clear and concise manner using the recommendations and guidelines discussed by Fowler and Stanwick (1995) and Horton (1994):

- Icons should look different from all unrelated icons.
- Related Icons should share a common thread.
- Icons should show the subject or object to which they are related.
- Metaphors should be used where objects cannot be shown.
- Existing symbols should be used wherever possible.
- Icons should be as simple as possible.

The size of the icons was set at 30 x 30 pixels, determined using the recommendations outlined by Fowler and Stanwick (1995). This size was thought to give the icons enough emphasis in terms of the overall screen display, without taking up too much room with respect to the Map Display. Lesser sizes (e.g. 20 x 20 pixels) were deemed too small, making the icons look fairly unimportant.

The GPS functions icon was designed as a pictorial representation of a satellite, as this seemed to be the most obvious object to associate with GPS. A yellow colour was chosen to make the icon stand out as, according to Sutcliffe (1995), yellow is excellent for highlighting symbols (Figure 5.26).

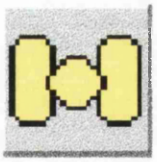


Figure 5.26 GPS Icon

The data storage icon was displayed as a symbol depicting a folder or paper file. Two shades of yellow were used to make the symbol stand out, and give the symbol depth.
This type of symbol exists in many computer packages to represent data storage and retrieval functions, and according to Fowler and Stanwick (1995) it is good design policy to re-use existing designs as many users will already understand their meaning (Figure 5.27).

Figure 5.27 Data Storage Icon

The icon representing the route planning functions was another associative icon. This was based on the idea of a route being a means of getting from A to B. Black had to be used for the line and text for contrast against the grey background. Red and blue start and end points were used in accordance with the guidelines laid down by Travis (1991) to make the icon visually more attractive (Figure 5.28).

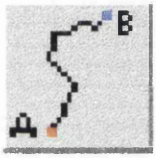


Figure 5.28 Route Planning Icon

The compass icon was uses an associative symbol. The icon depicts a compass needle and North point. This symbol has a very strong association with the compass, and the addition of the 'N' reinforces this. This follows the guidelines laid down by Fowler and Stanwick (1995) of using traditional images. Yellow was used to infill the needle so that it contrasted well with the black outline (Figure 5.29).

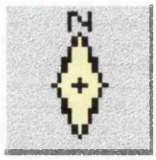


Figure 5.29 Compass Icon

The Zoom In icon was depicted symbolically as a magnifying glass as illustrated by Horton (1994). This symbol is used in many software packages to represent similar functions. The Plus '+' sign is used to show the image will enlarge through use of this command (Figure 5.30).

Figure 5.30 Zoom In Icon

The Zoom out command represents the opposite action to the Zoom in Command. Because the two functions are so closely related the same magnifying glass symbol was used but with a negative '-' sign to show the image will be reduced (Figure 5.31).

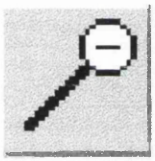


Figure 5.31 Zoom Out Icon

The Map Move icon pictorially depicts a map in two different places and then utilises arrows to symbolise the movement of the map from one location to the other. Bright red, blue and green are added to give the icon greater visual impact (Figure 5.32).

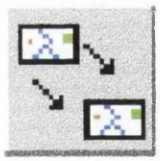


Figure 5.32 Map Move Icon

The concept of recentring proved difficult to symbolise. The icon depicts a target or gunsight. The user should associate this imagery with centring on a target, i.e. the user's position (Figure 5.33).

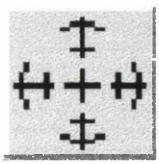


Figure 5.33 Recentre Icon

The symbol used on the Cursor Icon represented the shape of the actual cursor itself. According to Horton (1994) a black arrow is commonly associated with the selection of objects which is the main function of the cursor in the navigation environment (Figure 5.34).

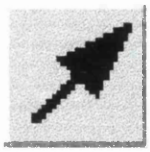


Figure 5.34 Cursor Icon

This final level one icon is the Help icon. This employs the query in the context of the user having questions of the system that can be answered by selecting this icon, as described by Horton (1994) (Figure 5.35). Once again such an icon is used in many existing software packages.

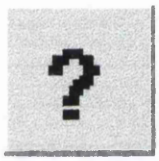


Figure 5.35 Help Icon

All these icons were located on the toolbar in a horizontal line. In addition to this, a status bar would also appear in the toolbar, displaying information such as co-ordinates, time and bearing etc. (Figure 5.36).

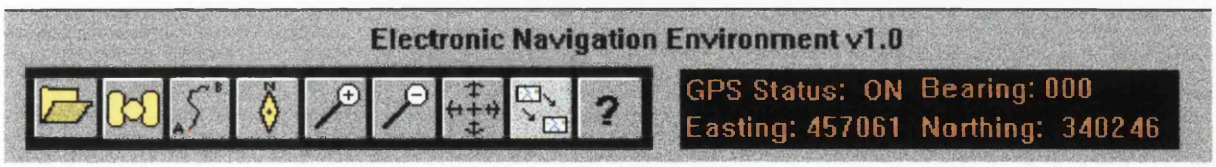


Figure 5.36 Toolbar

(iii) Level Two Representation

Pop-Up menus are ideal for Level Two representation. They can be used to display icons, text, command buttons and check boxes. This allows more information to be displayed and so functions can be explained more clearly than at Level One

The GPS control menu utilised Radio buttons, command buttons and a list box to display its options. The radio buttons were used to switch the GPS receiver on or off. A single button could have been used as an on/off combination but separate buttons provided more clarity. The buttons were coloured, with red signifying GPS Off and green signifying GPS On, as according to Bergum and Bergum (1981) people strongly associate these colours with these actions. The Fix Interval command was designed as a list box. A List Box allows a selectable list to be displayed to the user. The list was designed to contain preset intervals in seconds which the user could easily select. The Map Display Commands (Fixed Map Display, Moving Map Display and Trail) would use command buttons. Clicking on the relevant button would select the appropriate map display (Figure 5.37).

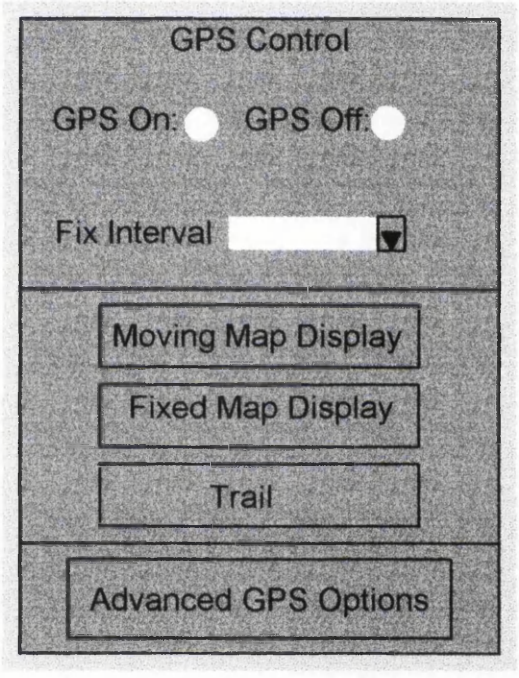


Figure 5.37 GPS Menu

The Data Storage and Retrieval Pop-Up Menu was designed to use icons and command buttons to display its options (Figure 5.38).

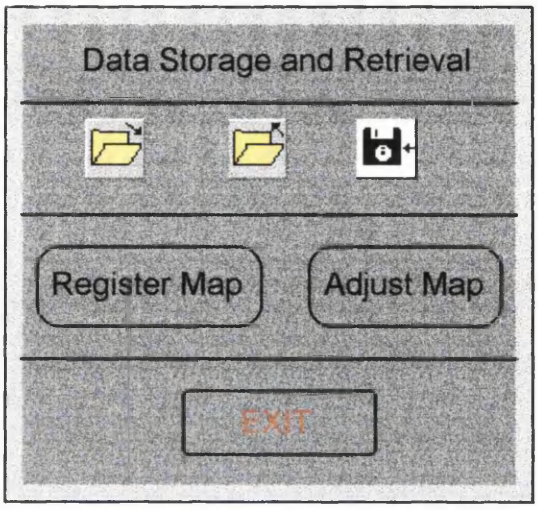


Figure 5.38 Data Storage and Retrieval Menu

The Open File and Close File icons used the same symbology as the Data Storage and Retrieval icon shown in Figure 5.27. However, these icons differed by using an arrow to explain each individual function as suggested by Horton (1994). The first icon shows Open File, with the black arrow representing the action of opening the file. The second icon shows Close File, with the arrow reversed. Save File was represented by a floppy disc with an arrow denoting the transfer of data onto the disc, as illustrated by Diaper and Winder (1987) (Figure 5.39).

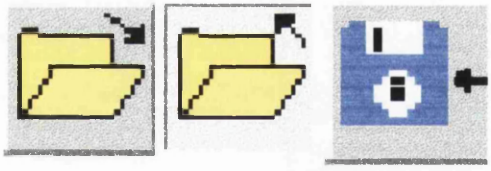


Figure 5.39 Data Storage and Retrieval Command Icons

The Map Set Up and Adjust were represented by Command Buttons, as these functions were seen as too complex to display using icons. Finally, the Exit command was to be displayed through the use of a Command Button. The text used was red in accordance with the guidelines laid down by Sutcliffe (1995), to illustrate that selecting this option will terminate the navigation software (Figure 5.38).

The Route Planning menu (Figure 5.40) used a series of command buttons, enhanced with graphics, as well as a list box and text window. Command buttons were used to display the plot route, plot waypoint, select, delete and measuring functions. Each was to be annotated

with the relevant command name, and a graphical symbol representing the function. The purpose of this was to make these commonly used functions as clear as possible, with the use of graphics adding colour to the buttons to make them stand out.

The plot waypoint button was annotated with the waypoint symbol that will be plotted on the map when this function is used. The plot route command was annotated with the line style used to depict a user defined route. The Select button was annotated with a representation of a selection box, and the delete command represented by a red cross (Horton, 1994). Finally Measure was symbolised by a pictorial symbol of a ruler. All these symbols were designed using the guide laid down by Sutcliffe (1988) whereby the command buttons and their related symbols should have a strong relationship with the actual task carried out.

List boxes were designed to display the various speeds at which the user may walk. Entering a speed and distance (via a distance list box) would lead to the output of journey time in the text window.

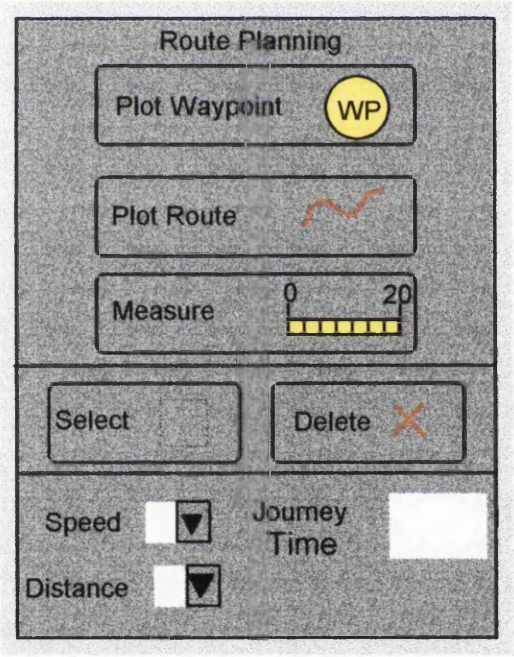


Figure 5.40 Route Planning Menu

The compass was to be represented in graphical form to give the user some familiarity between a magnetic compass and the electronic variation to be used by the software. Thus, a rotating needle was designed with north, south, east and west marked as in a conventional hand-held compass (Figure 5.41).

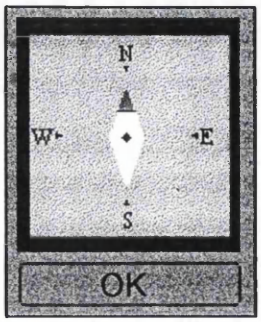


Figure 5.41 Compass Graphic

The Help Contents were designed on a white background as this has conventionally been used to represent help pages in the Windows environment. Each heading was represented in green text to signify that the text is help associated and selectable in accordance with the recommendations laid down by Travis (1991). Clicking on an item of text would activate the relevant help window (Figure 5.42). A pictorial design was added to enhance the visual impact of the menu.

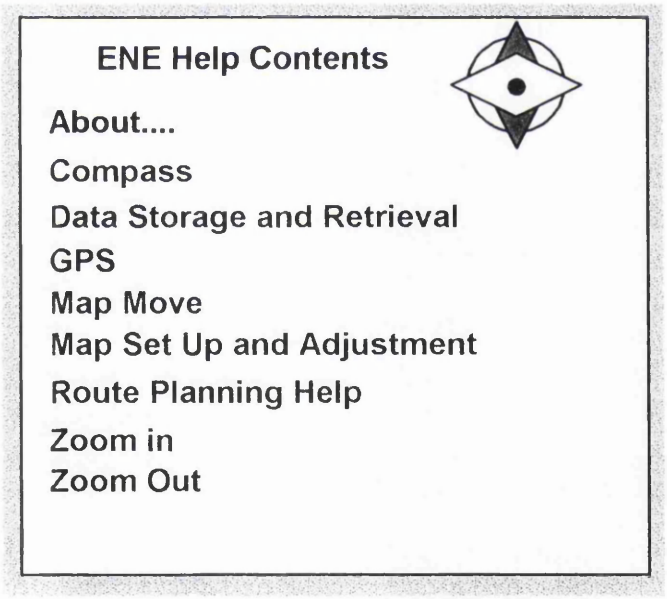


Figure 5.42 Help Contents

(iv) Level Three Representation

The Open and Save File menus would follow the convention used by almost all Windows-based packages. These were provided as pre-designed modules by the MapBasic software. The Adjustment and Register functions were also provided as pre-designed menus.

The help pop-up menus were again designed with a white background to denote the help option. Pictures, text and diagrams were to be used to describe each function available (Figure 5.43).

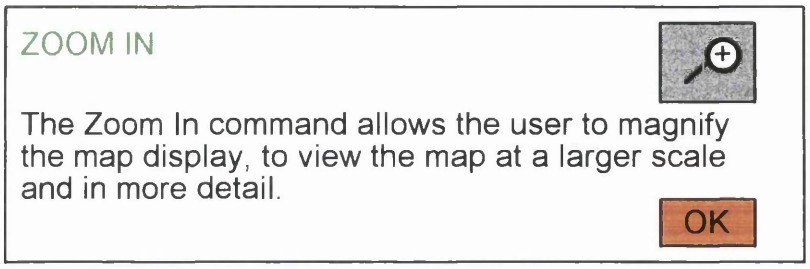


Figure 5.43 Zoom In Help Menu

The Advanced GPS functions were represented by the Trimble GPS Control Panel. This menu was pre-designed and required no further development.

5.5.4 Colour

Colour has been mentioned throughout this design model in connection with icon design and command representation. This section seeks to give a general overview of how colour was used in the Graphical User Interface. There are three main types of features within the interface that were coloured. Those in the Background, those in the Mid-ground and those in the Foreground. Background features are essentially the back drop to the interface or 'wallpaper'. Midground features are command buttons, icon buttons or status boxes. They are not recognised directly on their own, but by the Foreground features they hold. Foreground features are the most important features as they give access to the midground features; foreground features essentially involve text and graphical symbols or pictures. User interface features can be divided thus:

Background Features	**Mid-ground Features**	**Foreground Features**
Menu Boxes	Command Buttons	Menu Text
Tool Bar	Icon Buttons	Command Button Text / Graphics
Help Menus	Status Box	Icon Text / Graphics
		Status Box Text
		Map Display
		Help Text

The foreground colours have been discussed throughout the representation of options section, so only the Mid and Background colours will be discussed here.

The background features provide a backdrop to more important features and, according to Sutcliffe (1988) should be given unobtrusive or pale colours. To this end, a medium grey shade was most appropriate, as this does not attract attention to itself and is easy on the eye. Brighter colours were deemed too vibrant as they can strain the eye if used in large quantities as a background colour. Therefore, as noted by Shneiderman (1987) bright colours were used conservatively. White could also have been used as a background colour as it is neutral and provides good contrast with most other colours. However, in large quantities, especially on a computer screen, it can appear too bright compared with grey which can be warmer and more comfortable to look at for long periods.
A medium grey colour was used for the toolbar and Menu boxes, with the Help Boxes given a white background to differentiate between help functions and actual program functions.

Background Features:

 Menu Boxes, Toolbar: Medium Grey

 Help Menus: White

Mid-ground features should stand out from the background but not to the extent where they overwhelm foreground colours. Thus neutral colours were used once again, but darker or lighter shades were used to provide contrast and highlighting as outlined by Sutcliffe (1988). Alternatively the same colours could be used for back and foreground features but a black outline may be used to provide contrast, or 3D shading may make the feature stand out.

Mid-ground Features:

 Command Buttons, Icon Buttons: Medium Grey with 3D effec

 Status Box: Black

The colours selected to represent the Electronic Navigation Environment were by no means fixed, and were open to redesign if deemed unsuitable when coded.

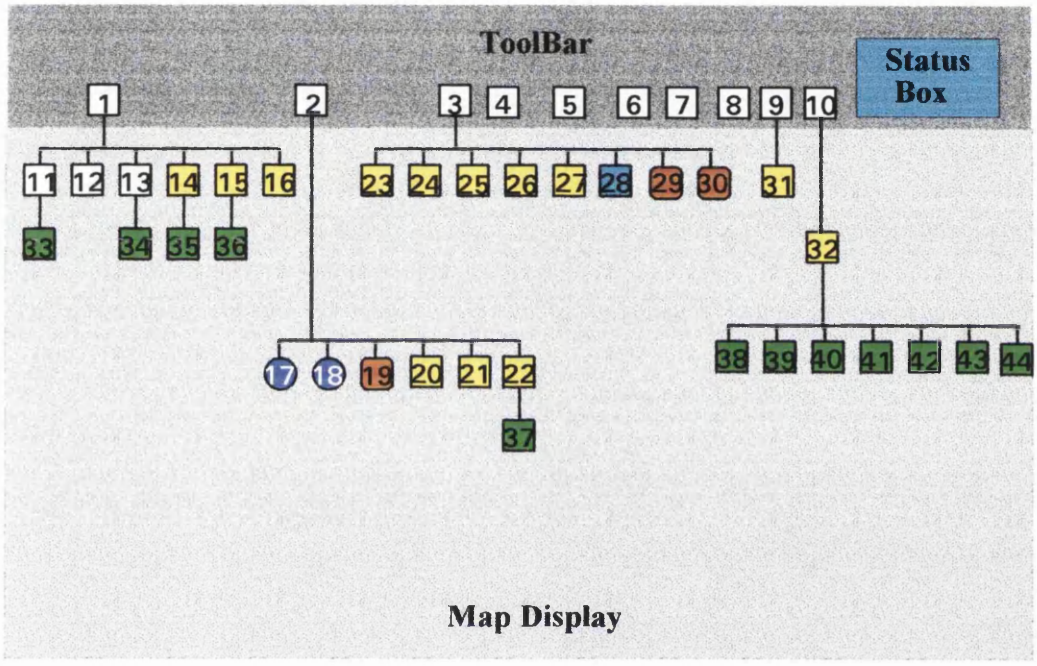
5.5.5 Help Prompts

The final stage of the System Image was to create the Help Prompts. These were designed as dialogue boxes. The precise dialogue boxes required were deemed easier to implement

and design as the system was being developed, as it was difficult at the design stage to envisage where they would be required.

5.5.6 Software User Interface Design Summary

The Software User Interface was to be coded using Visual Basic, and so it belonged to the Visual Basic program group. In summary, the it was designed as follows (Figure 5.44):



Key

- Icon
- Command Button
- Pop-Up Menu
- Text Box
- Radio Button
- List Box

(Numbers correspond to menu system.)

Figure 5.44 Software User Interface Design Summary

5.6 Electronic Navigation Environment Summary

This concludes the design of each module of the Electronic Navigation Environment. As can be seen from the design process, this basically involved two programs, one in MapBasic, and one in Visual Basic, and then linking them together via the Dynamic Data Exchange. The design of the navigation software and all of its modules can be summarised with a simple block diagram (Figure 5.45).

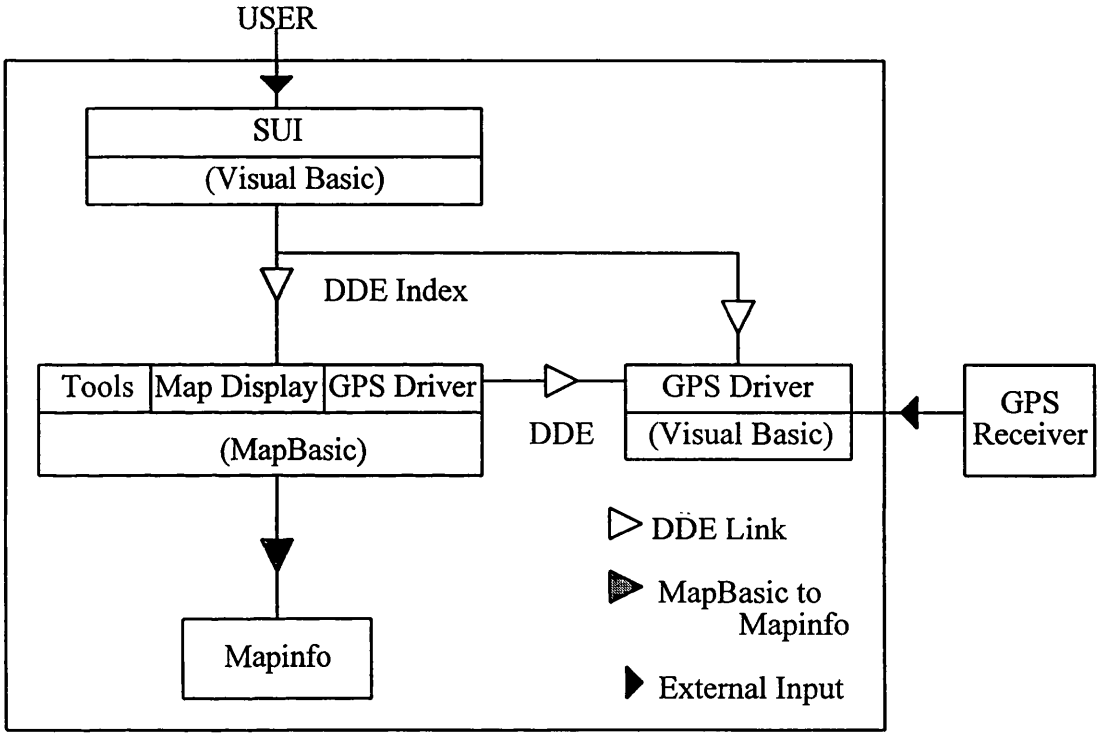


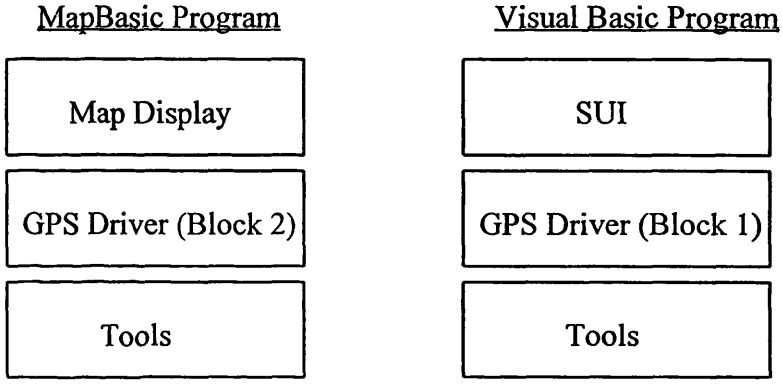
Figure 5.45 Electronic Navigation Environment Design Summary

The next stage was to code the programs and create the modules, based on the designs outlined in this chapter.

CHAPTER 6: SOFTWARE DEVELOPMENT

6.1 Introduction

This chapter explains and discusses the development, coding and linking of the Electronic Navigation Environment modules. The program groups discussed in the previous chapter were coded using the modular method as described by Brown and Sampson (1973), from the Map display, to GPS Driver, Tools and Software User Interface (SUI). At the end of the development stage, the program groups would contain the following elements:



6.2 Map Display Coding

The Map Display was essentially an initialisation phase for the Electronic Navigation Environment. As had been established in the design stage, the following procedures were required:

- Main
- Set Window Size
- Hide Menu Bar

The first procedure to be coded determined the map window size. This was named 'WindowSet' and was called from the Main procedure:

```
Sub WindowSet
    StatusBar Hide
    Set Window WIN_MAPINFO
        Position (0,18)
        Width 169.5
        Height 109
End Sub
```

The first command removed the status bar from the map window. The 'Set Window' command positioned and sized the window using three parameters: position, height and width. The units used within these parameters had to be established as millimetres, centimetres or inches. The units were set to millimetres in the Main procedure, which set the units for the whole program. If the units were set in the WindowSet procedure, they would only remain set while the procedure was being executed i.e. setting the units locally instead of globally. The window was thus positioned and sized in millimetres to fit the specified area of screen. This procedure, when executed, operated as planned, but some fine tuning was required to ensure the window size fitted the screen exactly.

The next procedure involved removing the MapInfo menu bar using 'Menu Bar Hide':

```
Sub AlterMenuBar
    Menu Bar Hide
End Sub
```

This procedure operated successfully when executed at a modular level.
Finally, the Main procedure was coded. This involved the following steps:

Main Procedure
- Set Units
- Call WindowSet
- Execute Visual Basic Group
- Set ProgressBars Off
- Hide Main Buttonpad
- Hide Tools Buttonpad
- Hide Drawing Buttonpad
- Call AlterMenuBar

End Sub

The Map Display was tested by execution using the method described by Pritchard (1988), to ensure the correct output was achieved and the program terminated correctly. The map display window was positioned correctly and the MapInfo buttonpads were removed from the display. Progress bars were also disabled. The Map Display Module also had to launch the Visual Basic program group. Even though at this stage no Visual Basic program existed, this was tested using a dummy program and proved operational. The ouput, therefore, was as expected.

The program termination, however, did not operate as planned. It was clear that once the program had called its procedures it terminated. This meant that the MapBasic group, of which this was the basis, would not be able to provide further functions. A method was required to prevent the program from terminating at this stage.
It was decided to create a buttonpad that would sit on the screen but be invisible to the user. This would prevent the program from terminating as it would wait for a response from the user and thus the MapBasic Group would remain in use. A new procedure was added to the map display to carry this out - 'NewToolBar'

```
Sub NewToolBar
        Create Toolbar to Call ButtonResponse
                Hide Toolbar
                Disable Toolbar
End Sub
```

Disabling the button pad was very important so that if the user accidentally selected it, nothing would happen. This procedure prevented the program from terminating and provided a basis on which to build the MapBasic group of the Electronic Navigation Environment. In summary, the Map Display consisted of the following procedures (Figure 6.1):

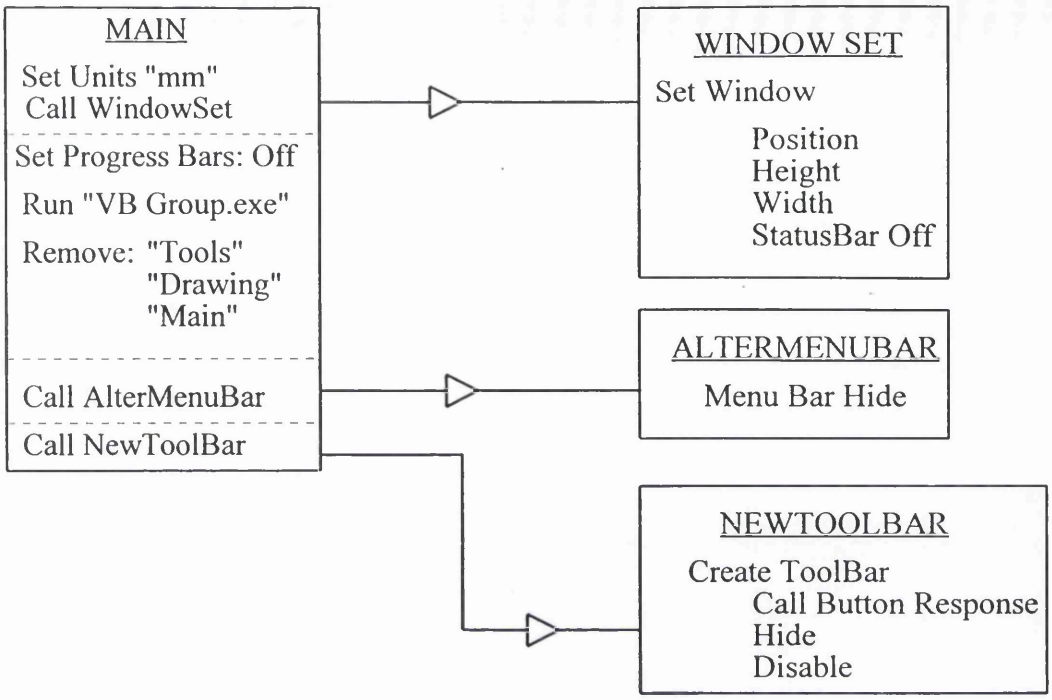
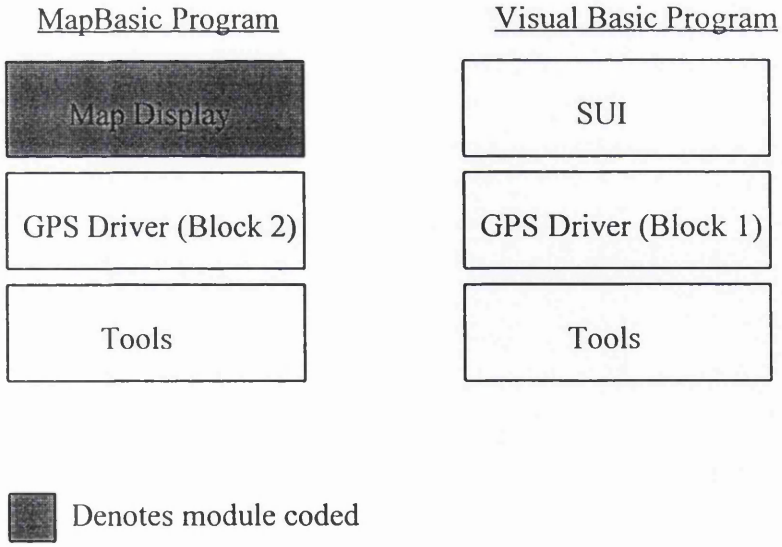


Figure 6.1 Map Display Module

This completed the first Electronic Navigation Environment module:



6.3 GPS Driver

The GPS Driver was coded in two Blocks. Block 1 was coded first in Visual Basic, followed by Block 2 in MapBasic. These Blocks were then linked using Dynamic Data Exchange.

6.3.1 Block 1 Development

With Block 1 being developed in Visual Basic, the first stage was visual programming as described by Gurewich and Gurewich (1995). A form was created called the co-ordinate display window to hold co-ordinate information, before being sent using DDE to Block 2. Such a window was required as data could not be sent from Visual Basic code to MapBasic code using DDE. Any data to be sent had to be written to a Visual Basic form first. The co-ordinate display window contained two text boxes, one for the Easting co-ordinate value and one for the Northing (Figure 6.2).

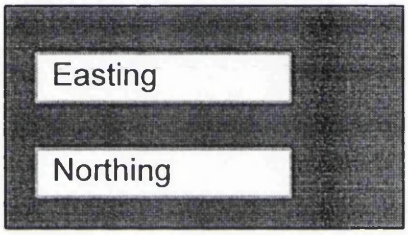


Figure 6.2 Visual Programming of Co-ordinate Display Window

This window was hidden from view during operation so as not to impair the user's view of the Map Display. With the visual programming completed, the Block 1 code was attached.

Coding Block 1 involved three stages, employing the commands and structures described by Trimble Navigation (1994b). When the GPS Driver was switched on, the co-ordinate display window form was loaded, which invoked the initialisation process. Once completed, the keydown procedure was called each time data was received from the GPS receiver. This procedure involved three stages: data conversion, transformation and fix interval (Figure 6.3).

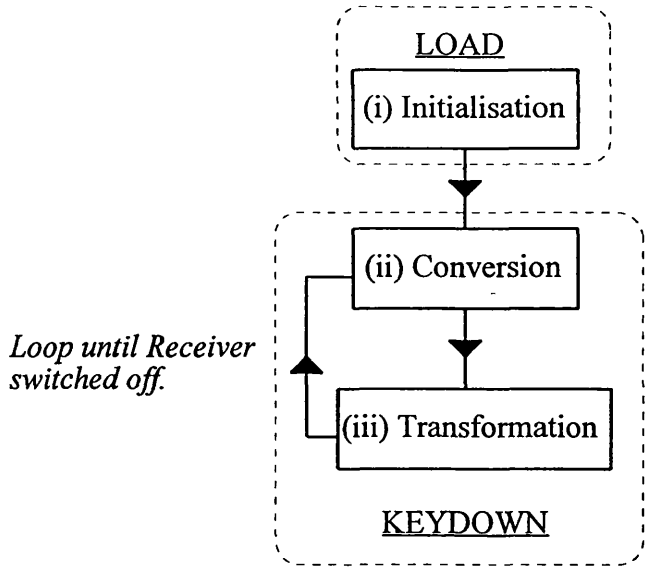


Figure 6.3 Block 1 Structure and Flow Diagram

(i) Initialisation

Initialisation was carried out when the GPS receiver was switched on, via the Software User Interface. This process was located in the LOAD procedure of the co-ordinate display window, and involved the following steps:

(1) Register GPS Driver with Dynamic Link Library
(2) Link Receiver to the TSIP/ASCII Conversion procedure.
(3) Send Status Information to Status Box

Stage (1) was written using the 'wglInstall()' command. This linked the GPS receiver to the GPS Driver, via the Dynamic Link Library (DLL), and loaded the Trimble GPS Control Panel by default:

```
nRet = wglInstallEx(ByVal 0&, SW_MINIMIZE)
```

The above code linked the receiver and driver using DLL defined commands. It also minimised the GPS control panel so that it ddid not obscure the map display.

Stage (2) was coded out using the wglHookSuperPosition(PosData) command. This assigned any data from the GPS receiver to a variable(PosData). This data was then sent to the KEYDOWN procedure:

PosHookMsgs.message = WM_KEYDOWN
(Tells the program to send all data recieved to the Keydown procedure)
PosReqHandle = wglHookSuperPosition (PosData)
(Assigns GPS data to variable (PosData))

Stage (3) notifies the user via the Status box in the SUI that the GPS receiver has been switched on, by sending a text message to the box. With the receiver and driver initialised, it is ready to process data.

(ii) TSIP to ASCII Conversion
TSIP to ASCII conversion was carried out in the keydown procedure of the co-ordinate display window, and required the execution of three stages:

(1) Get TSIP binary data
(2) Extract Position Information from binary data
(3) Format Position Information for output as Degrees, Minutes and Seconds

Stage (1) was achieved using the 'wglGetPosBinData()' command, which called the binary data from the DLL and stored it in a variable. Once the binary data had been stored, the positional information had to be extracted in Stage (2). This was achieved using 'wgldGetPosDataLLA(PosData,PosDegMin)'. This function required the use of two variables. Variable "PosData" corresponded to that used in Stage (1), (i.e. it was the variable used to store the binary data). Variable "PosDegMin" was where the extracted positional information was to be stored. Variable PosDegMin was of a special type, created by the DLL. This variable allowed structured data to be stored, which could then be extracted by specifying the parts required. The data was stored in degrees and minutes (Figure 6.4).

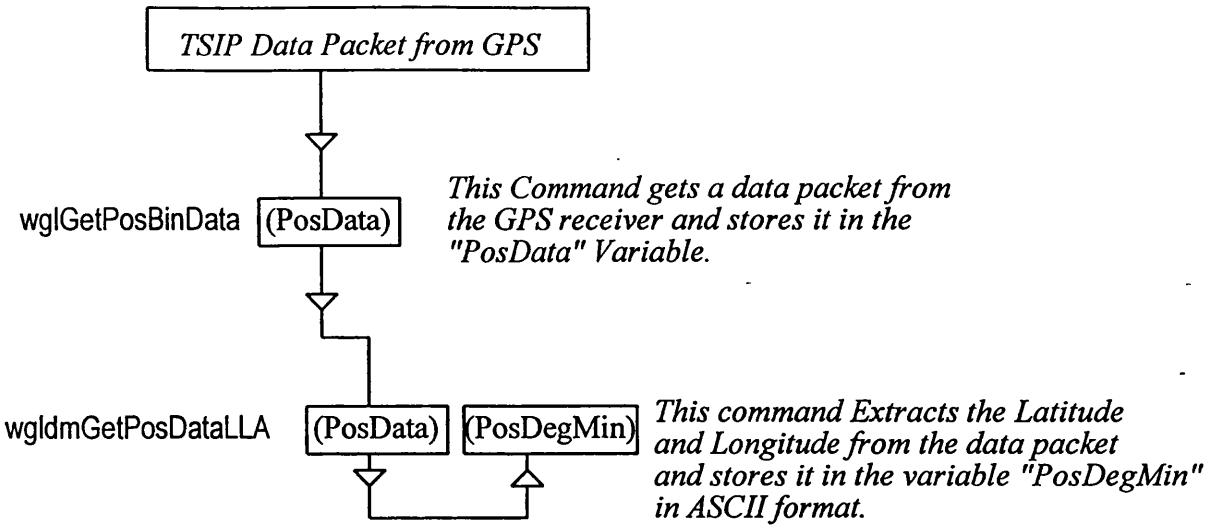


Figure 6.4 Binary - ASCII Conversion Code

Stage (3) formated the information by storing the Latitude and Longitude values from variable "PosDegMin" in separate variables:

```
LongitudeDegrees = (PosDegMin.Longitude.Degrees)
LongitudeMinutes = (PosDegMin.Longitude.Minutes)
LongitudeSeconds = (PosDegMin.Longitude.Minutes) - (LongitudeMinutes*60)
LatitudeDegrees = (PosDegMin.Latitude.Degrees)
LatitudeMinutes = (PosDegMin.Latitude.Minutes)
LatitudeSeconds = (PosDegMin.Latitude.Minutes) - (LatitudeMinutes*60)
```

At this point the data was ASCII formatted, but degrees and minutes were stored and extracted separately from the data packet. To gain the true Latitude and Longitude these values had to be combined:

```
Longitude = LongitudeDegrees + ((( LongitudeMinutes*60)+ LongitudeSeconds) / 3600)
Latitude = LatitudeDegrees + ((( LatitudeMinutes*60)+ LatitudeSeconds) / 3600)
```

With these three steps completed the data had been converted to single Latitude and Longitude values and stored in two separate variables. The next stage in Block 1 development was to transform these values to Ordnance Survey National Grid (OSNG) co-ordinates.

(iii) World Geodetic System 1984 to Ordnance Survey National Grid Transformation

Coding the transformation involved entering the relevant equations and functions and ensuring that the correct results were obtained at each stage of the transformation before

proceeding. The transformation was set up during coding to accept test co-ordinates from a comma delimited ASCII file, which were transformed and output to a text file for inspection (Figure 6.5).

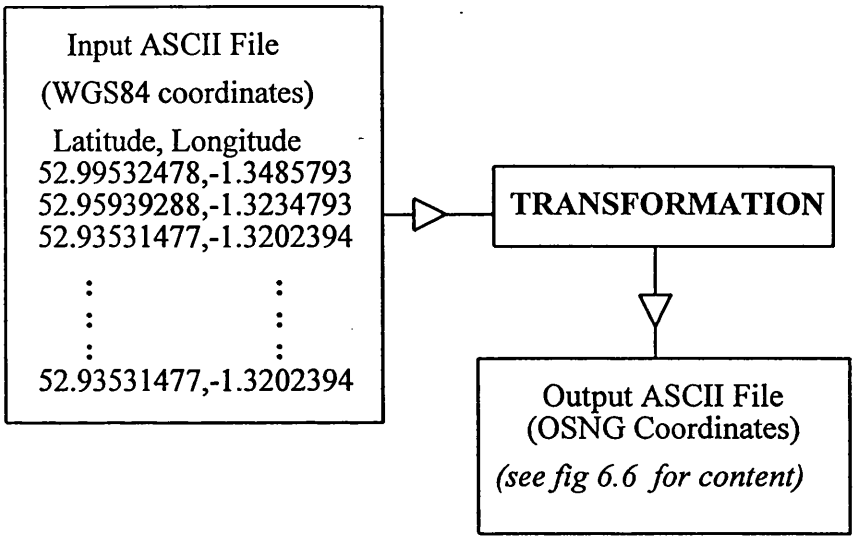


Figure 6.5 Transformation Input/ Output (Modular Testing)

The complex nature of some of the equations meant that care had to be taken to ensure they functioned correctly. This required step-by-step examples with which to compare the code. A worked example was supplied in the Ordnance Survey documentation and so the results of each equation were written to the output file for visual inspection using the approach described by Fagan (1976) (Figure 6.6). The Desk Checking method, as described by Pritchard (1988), was also used as a means to test the code by hand, and compare it to hand calculated examples. These approaches were seen as the most effective methods of trapping any errors that occurred during the transformation.

Sample output from error trapping file T4.DAT:

```
"Eastings","Northings"
"Latitude",51.9761,"Longitude",-1.606912639
"P",6.86066869742554E-03
"v",6388878.12981108
"n2",2.55726052271221E-03
"M",330926.545378067
"I",230926.545378067
"II",1550097.10980232
"III",166040.725742943
"IIIIA",-19330.206812275
"IV",3935485.85432522
"V",-157512.566581951
"VI",-103738.8726742
"OSGRS80",427000.01374388,230999.506917975
"east_index",1.22000003926823,"north_index",.6599986
"east_index (after fix)",1,"north_index",0
"s0",96,"s1",102,"s2",105,"s3",96
"xCell",350000,"yCell",0
"t",.220000039268229,"u",.659998591194215
"se",97.7555993835485
"NORTH INDEX",0
"s0",-80,"s1",-82,"s2",-78,"s3",-75
"xCell",350000,"yCell",0
"t",.220279340980753,"u",.659998591194215
"s",-77.2859497807069
"Ordnance Survey National Grid"
427097.769343264,230922.220968194
```

(These parameters were compared with the sample data supplied with the equations or with hand calculated examples in order to determine whether the correct values were being calculated by the code.)

Figure 6.6 Error Trapping

The main logic problem encountered whilst programming was setting up and accessing the matrices required to apply the shifts from OSGRS80 to OSNG. Visual Basic does not seem to be equipped to handle matrices, therefore the alternative was to set up a database with the shifts as fields, which Visual Basic is ideally equipped to carry out. However, this solution would utilise too much memory compared with the use of two-dimensional linear arrays. An array is a single string of data values which can be accessed in a linear fashion. Each element in the array would represent a shift parameter. Each array represented a row in the matrix and a series of conditions had to be set up to ensure the correct array was accessed:

Each row of the matrix was set up as a linear array
i.e.

L1(99,104,107)
L2(93,99,106)
: :
: :
L5(92,96,102)

OSGRS80 N		East shift (m)	
1400000	99	104	107
1050000	93	99	106
700000	85	97	108
350000	89	96	105
0	92	96	102
	0	350000	700000
		OSGRS80 E	

A sequence of If...then statements were then used to determine which array should be accessed, based on the east, north shift indexes:

```
If north_index = 0 Then
   s0 = L1(east_index)
ElseIf north_index = 1 Then
   s0 = L2(east_index)
ElseIf north_index = 2 Then
   s0 = L3(east_index)
ElseIf north_index = 3 Then
   s0 = L4(east_index)
End If
  :
  :
  :
  :
's3 parameters
If s0 = L1(east_index) Then
   s3 = L2(east_index)
ElseIf s0 = L2(east_index) Then
   s3 = L3(east_index)
ElseIf s0 = L3(east_index) Then
   s3 = L4(east_index)
ElseIf s0 = L4(east_index) Then
   s3 = L5(east_index)
End If
```

Once each stage of the transformation was operational, the whole program was run and the resultant data analysed to detect any run-time errors. The following errors were detected during this process.

The resultant co-ordinate value did not match the actual value when read from a data file. This was due to the Latitude value from the data file being only partially read into the program, with the last three decimal numbers missing. This was due to the variables which hold these co-ordinate values in the program having the incorrect type definition.

The solution to the problem was to change the type definition of these variables from single type to double type.

Another problem that arose was the access of the incorrect arrays when calculating the eastings and northings shifts. This occurred when the North index = 1 and East index = 0. The error was due to the fact that when the index was calculated, real numbers were not rounded down at all times. When numbers were rounded up the wrong index was calculated. This problem was solved using the Fix function.

This error and its solution is best explained using an example:

- East and North indexes calculated to 1.8,1.9 respectively
- Only integer values are required so these numbers must be truncated to give 1,1
- Visual Basic will round numbers up by default if an integer value is asked from a real number, giving: 2,2
- The Fix function truncates numbers, effectively rounding down

Once all errors had been removed from the program, test data was run through and the results analysed to ensure the original specifications of the transformation were being met (see Chapter 7: Testing, Results and Analysis).

(iv) Fix Interval

The final coding in Block 1 involved the fix interval loop. The original plan for the fix interval was to introduce a loop in the program that would count the number of seconds corresponding to the fix interval and then proceed to transform the co-ordinates. However, there were several problems with this method when implemented.

When the fix interval was tested, the program froze when accessing the time loop (Figure 6.7). It was clear that such a loop could not be used in real time as it always froze, even though the loop collected and processed the data. The second problem was that even if the loop could operate, it would dominate processing and freeze the cursor, preventing the user from carrying out other tasks whilst the GPS Driver was operational.

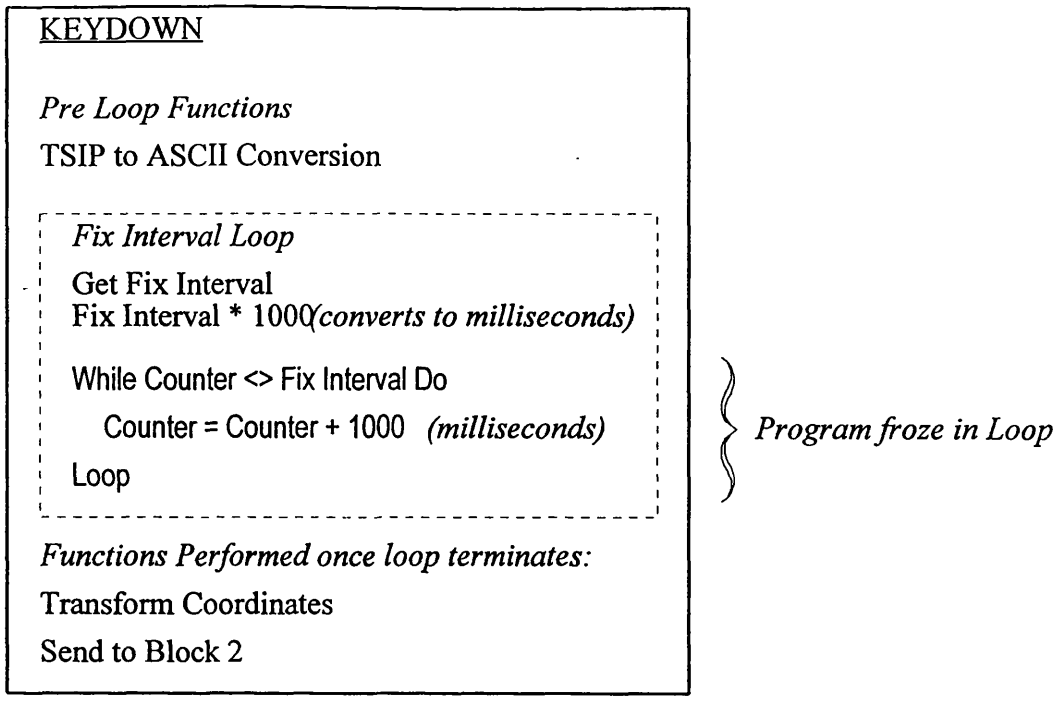
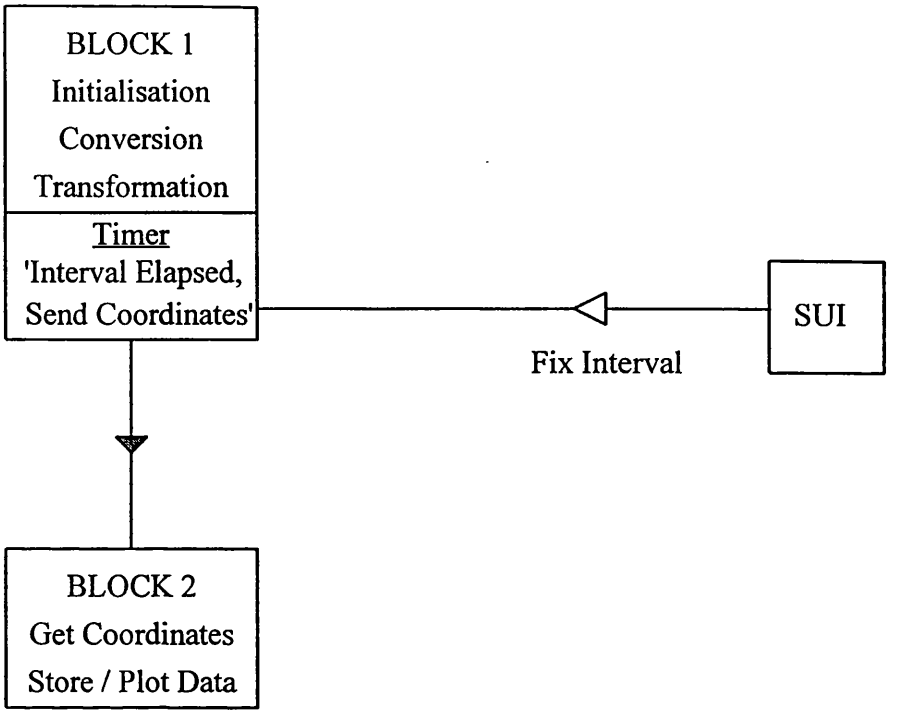


Figure 6.7 Fix Interval Loop

The solution was to use the timer function in Visual Basic which is described by Gurewich and Gurewich (1995). This function is specially designed to pause programs for a specified length of time, allowing other tasks to be carried out whilst the timer is counting. This function was ideally suited to the task, and would have been used initially if known at design time.

Coding the timer was simple, with the code to send co-ordinates to Block 2 being placed in the timer procedure. This meant that after the specified interval, the timer would send the message to Block 2 to start storing and plotting (Figure 6.8)



When Interval elapses on time coordinates are sent to BLOCK 2.

Figure 6.8 Timer Procedure

Once the fix interval had been coded, it was inserted into the GPS driver in the KEYDOWN procedure, along with the transformation. The transformed co-ordinates were written to the co-ordinate display window (Figure 6.9).

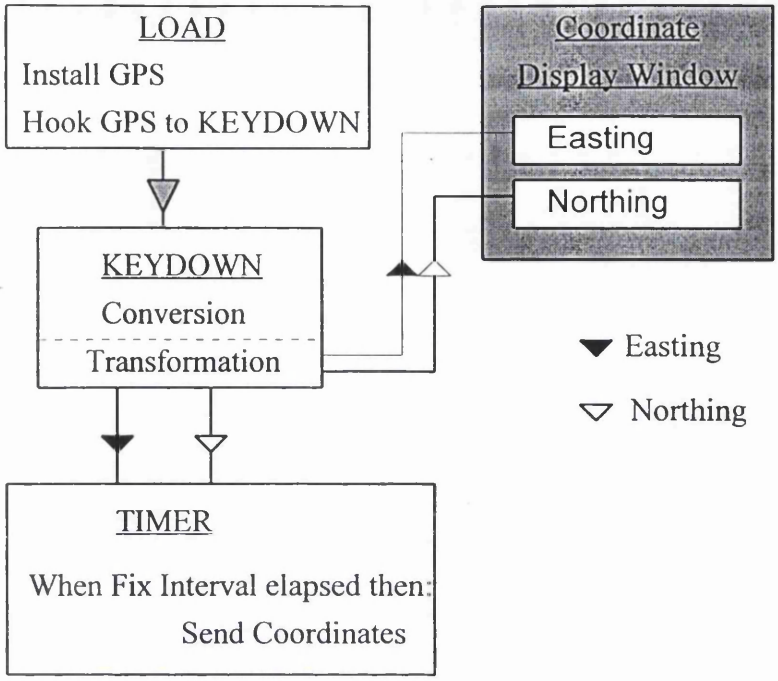
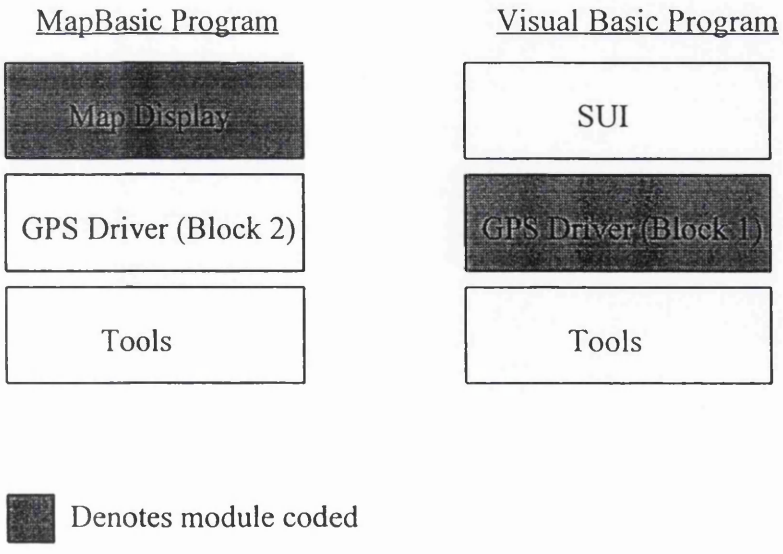


Figure 6.9 Block 1 Schematic

This concluded the coding of Block 1, with the exception of the Dynamic Data Exchange link to Block 2. This was not coded until Block 2 had been created, and so is discussed after Block 2 coding. Block 1 of the GPS Driver was the first piece of module to be inserted into the Visual Basic Group. The elements coded thus far were:



6.3.2 Block 2 Development

The development of Block 2 involved two stages: Data Storage and Graphical Plotting.

6.3.2.1 Data Storage

Data Storage involved setting up a database within MapInfo to store positional information. If this information were to be plotted on the map display the database had to be made 'mappable' (MapInfo Corporation, 1994b). In order to achieve these two aims the 'AddLayer' procedure was created. This involved the following steps:

(1) Create Database
(2) Create Map Layer for database
(3) Add Map Layer to Map Display
(4) Make Map Layer Editable

In order to carry out step (1), the 'Create Table' command was used. This created a database, requiring the names and type of any fields within the database, followed by its format and a path to save:

```
Create Table POSITION |      (This code creates a database called (Easting
float,Northing float)  |      POSITION with Easting and Northing
                       |      fields, of type floating point.)

                                (This code defines the path through
File "C:\MapInfo\OS\position"  |  which the database should be saved
Type NATIVE                    |  and its format, in this case 'Native'
                               |  (MapInfo format)).
```

Step (2) required the creation of a map layer for plotting. This was achieved using the 'Create Map' command:

```
Create Map
        For POSITION
        CoordSys NonEarth
        Units "m"
        Bounds(0,0) (600000,800000)
```

As stated by the (MapInfo Corporation, 1994c) a co-ordinate system must be defined. This ensured that the POSITION map layer coincided with the map displayed by the navigation environment. Failure to carry out this procedure would have meant that the points would

have been plotted in a default co-ordinate system, causing considerable error in plotting accuracy (Figure 6.10).

If no coordinate and projection system is defined, then the two layers will not coincide, leading to plotting inaccuracies

If both coordinates and Projection coincide, coordinates will be plotted correctly

Figure 6.10 Co-ordinate Definition

Step (3) added the map layer to the map display using the following code:

```
Add Map Layer POSITION
```

Finally, Step (4) switched the layer to editable mode so information could be plotted graphically:

```
Set Map
        Layer POSITION
                Editable On
```

Once the mappable database had been created, then position could be plotted graphically.

6.3.2.2 Graphical Plotting

A number of options were presented when plotting graphically, such as the use of a fixed or moving map display, or whether previous positions wiere to be shown or removed. A procedure also had to be written to stop the plotting process. The following procedures were coded to present these options to the user.

(i) Fixed Map Display

The aim of this procedure was to plot the position of the user on a base map, using data stored in the POSITION database. This particular procedure achieved this aim utilising a fixed map, moving point solution, so that as the user moves, their position moves across

the base map. In order to accomplish this solution the program was required to carry out the following tasks.

(1) Insert Easting, Northing into POSITION
(2) Get Easting, Northing from POSITION
(3) Plot Point (Easting,Northing)

Stage (1) involved storing the co-ordinate values into the POSITION database. This was achieved using the 'Insert' command, followed by the values to be entered:

```
Insert Into POSITION
        Values(Easting,Northing)
```

The co-ordinate values had to be entered into the database before plotting as they were received from Block 1 as a string, and in order to plot the point floating point values were required. Writing the values to POSITION and then reading from the database allowed the values to be converted to floating point. Coordinates could then be plotted using the 'Create Point' command and 'Make symbol' command:

```
Fetch First from POSITION
(this sets a pointer to the last entry in the database)
  Create Point
        (East,North)
(this makes a point object from the co-ordinate values)
Symbol MakeSymbol(33, Magenta, 12)
(this makes a plottable symbol, to the specifications set out in GPS Driver design.
The point is then plotted on the Map Display.)
```

(ii) Moving Map Display

The aim of this procedure was to plot the position of the user in the centre of the map display, and move the map around this fixed point. In order to achieve this the following steps were carried out:

(1) Insert Easting, Northing into POSITION database
(2) Get Easting, Northing from POSITION
(3) Determine Magnitude and Direction of Map movement
(4) Center / Pan Map on (Easting,Northing)
(5) Plot Point (Easting,Northing)

Steps (1),(2) and (4) were coded in the same manner as those steps used in the 'Fixed' procedure. The magnitude and direction of the map movement was determined using the following code:

```
dE = prevE - Easting
dN = prevN - Northing

If dE < 0 then
  Set Map Pan dE Units "m" west
          CoordSys Earth
          Projection 8,7,"m",-2,49,0.9996012717,400000,-100000
Else
  Set Map Pan dE Units "m" east
          CoordSys Earth
          Projection 8,7,"m",-2,49,0.9996012717,400000,-100000
End If
If dN < 0 then
  Set Map Pan dN Units "m" south
          CoordSys Earth
          Projection 8,7,"m",-2,49,0.9996012717,400000,-100000
Else
  Set Map Pan dN Units "m" north
          CoordSys Earth
          Projection 8,7,"m",-2,49,0.9996012717,400000,-100000
End If
```

The magnitude of movement was found by calculating the difference between previous and present co-ordinate values (dE and dN). The direction was found by determining whether these differences were positive or negative values. This was achieved by first calculating the dE and dN values and then using If....Else statements to determine whether the values were greater or less than zero. If dE and dN were equal to zero, then the map remained stationary.

The actual movement of the map was carried out using the 'Pan' statement, which, it was thought, would scroll the map on-screen in a similar fashion to scroll bars. However, when the procedure was tested it redrew the map instead of panning it, causing the display to 'flash' as the map was redrawn, which created a lack of continuity in the movement of the map. There was no alternative solution to scrolling the map and it was therefore decided to simplify the procedure by centring the map on each fix.

The map was centred on the co-ordinate values using the 'Set Map' command, followed by the sub-command 'Centre':

```
Set Map
  Centre (East,North)
```

This command redrew the map display with the co-ordinate values East, North as its centre.

(iii) Trail

The final plotting option to be coded was whether to display previous positions on the map display or delete them. It was decided that the user would select this option via the Software User Interface (SUI), which would send an index response via the DDE to the MapBasic GPS Driver Group.

If the trail option was selected then this sent a DDE response of "27" to the remote message handler of the MapBasic Group. If such a response was received, the trail procedure would be called and the trail variable set to one. If no response was received, the trail variable would default to 0 (this is set in the Main procedure). A second procedure was then called, which uses the following If...Then statement:

```
If Trail = 0 Then
        Call DelPoint
ElseIf Trail = 1 Then
End If
```

If the trail value was zero, then points would be deleted from the screen by calling the 'DelPoint' procedure. This operated by setting a pointer to the last insertion in the database, and then deleting it:

(1) Set Row identifier
(2) Delete From POSITION where ROWID = i

The Row identifier was set to zero in the main procedure during initialisation. When the 'DelPoint' procedure was called, this value was increased by two, to set it at the location of the co-ordinates corresponding to the graphical object to be deleted in the database (Figure 6.11).

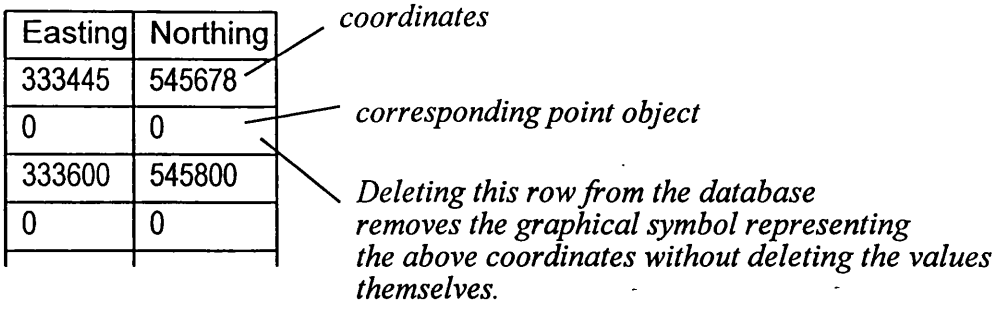


Figure 6.11 Point Deletion from the Database

(iv) GPSOff

The final procedure that needed to be created in Block 2 was the 'GPSOff' procedure. This procedure had to carry out the following tasks:

(1) End the Link between GPS DLL and GPS Driver Window
(2) Close the Trimble GPS Control Panel
(3) Send DDE message to MapBasic group to terminate MapBasic-VisualBasic DDE link

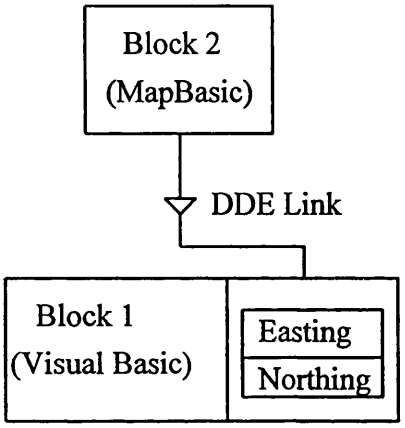
The user would switch the GPS Receiver and driver off via the SUI. When this option was selected it would send a response of "22" to the MapBasic group and unload the GPS Driver Window. This invoked the unload procedure which in turn shut down the Trimble GPS Control Panel and switched off the GPS Receiver.
In the MapBasic Group, when the GPS Driver was switched off the DDE links were to be terminated. This was achieved using the 'DDETerminateAll' Command. The DDE channel identifier i_chan was reset, so that the GPS Driver could be reused without the Electronic Navigation Environment being terminated. This procedure involved the following code:

```
DDETerminateAll
i_chan = 0
```

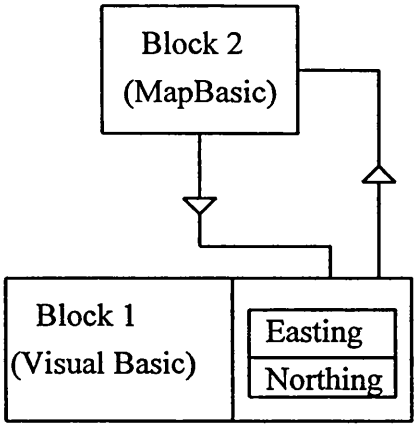
Once these procedures had been completed, they were inserted into the MapBasic Group. These procedures when compiled would not function unless linked to Block 2. Therefore the next stage was to link Block 1 and Block 2 of the GPS Driver.

6.3.3 Block 1 to Block 2 Dynamic Data Exchange Conversation

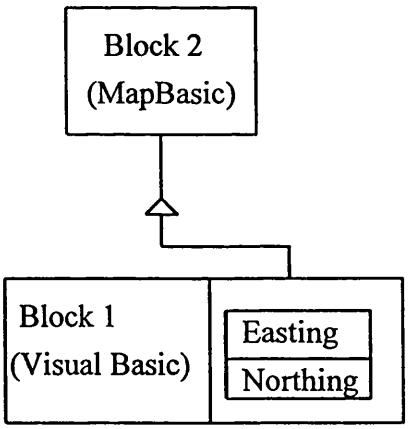
In the original design of the GPS Driver DDE link, co-ordinates were sent directly from Block 1 to Block 2. However, it soon became apparent that such a simple link would not be sufficient to transfer co-ordinates in real-time.



1) DDE Link established with Block 2 as client and Block 1 as server.



2) Block 2 checks for coordinates in the Block 1 coordinate display window using a loop.



3) 'Pull' coordinates from Block 1 to Block 2 for storage and plotting

Figure 6.12 Block 2 to Block 1 Dynamic Data Exchange_ Link

This method would have clashed with the indexing system used by the SUI. If co-ordinate values were sent via the DDE index, the user would have been unable to select any tools at the time of co-ordinate transmission. The index link would also be in danger of being overcrowded, trying to send more than one piece of information at one time. This would have lead to instability in the link. Alternative solutions had to be investigated.

The main alternative was to establish a link with Block 2 (in the MapBasic Group) as the client and Block 1 as the server. Block 2 would wait until co-ordinates were present in the co-ordinate display window of Block 1 and then 'pull' them through for storage and plotting (Figure 6.12).

This approach would have been impractical as it required a loop, the running of which would have dominated processing, disabling the mouse cursor and hindering the users ability to perform any other functions whilst the GPS driver was running. This would have included turning the GPS Driver off, so the process would have run indefinitely.

This problem was solved by employing the following method. When a co-ordinate value was received by the GPS driver, an index number "99" was sent to the Map Basic group via the DDE index. This operated in the normal manner, activating the Remote Message Handler which then called the relevant procedure. The procedure called differed from other DDE activated procedures in that it then initiated a DDE conversation, which "pulled" the co-ordinate data from Block 1 (Visual Basic Group) to Block 2 (MapBasic Group). A further procedure was called once the co-ordinates had been obtained in order to plot the position on the digital map displayed in MapInfo (Figure 6.13).

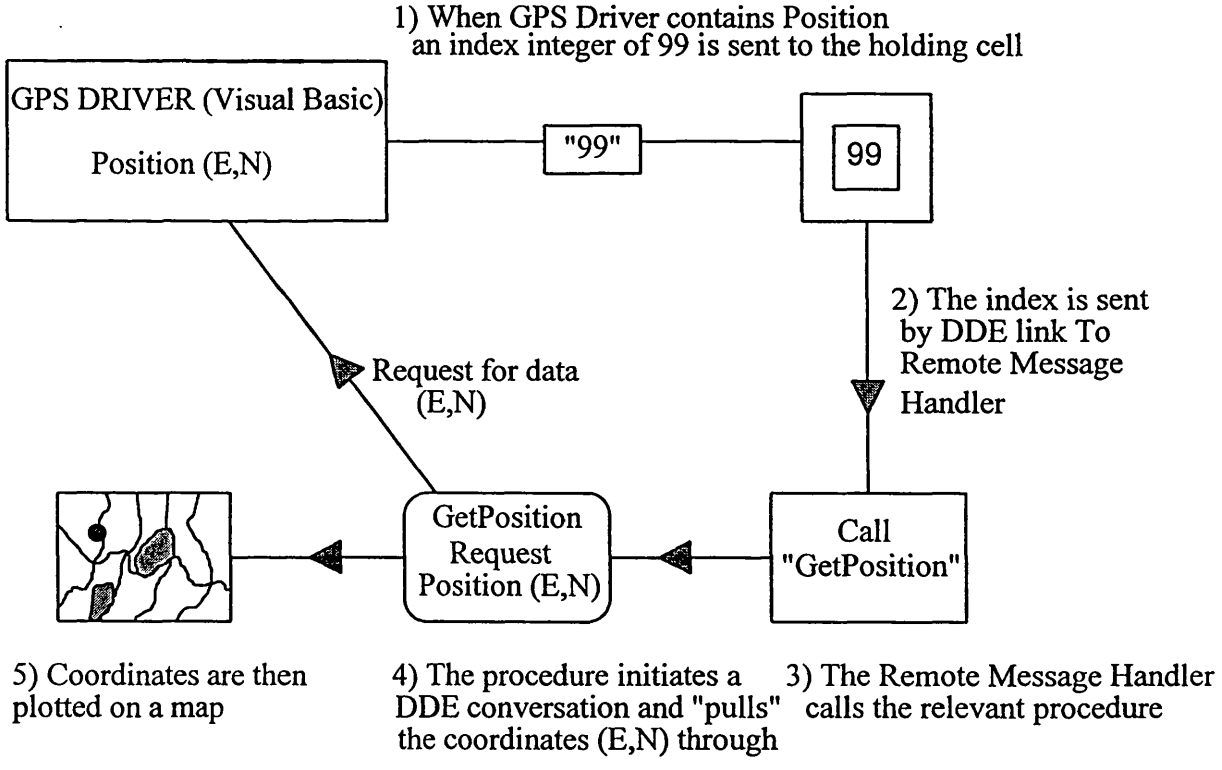


Figure 6.13 GPS Dynamic Data Exchange Link

Coding this method used the following procedures, which not only had to plot position, but use the plotting method selected by the user (Figure 6.14):

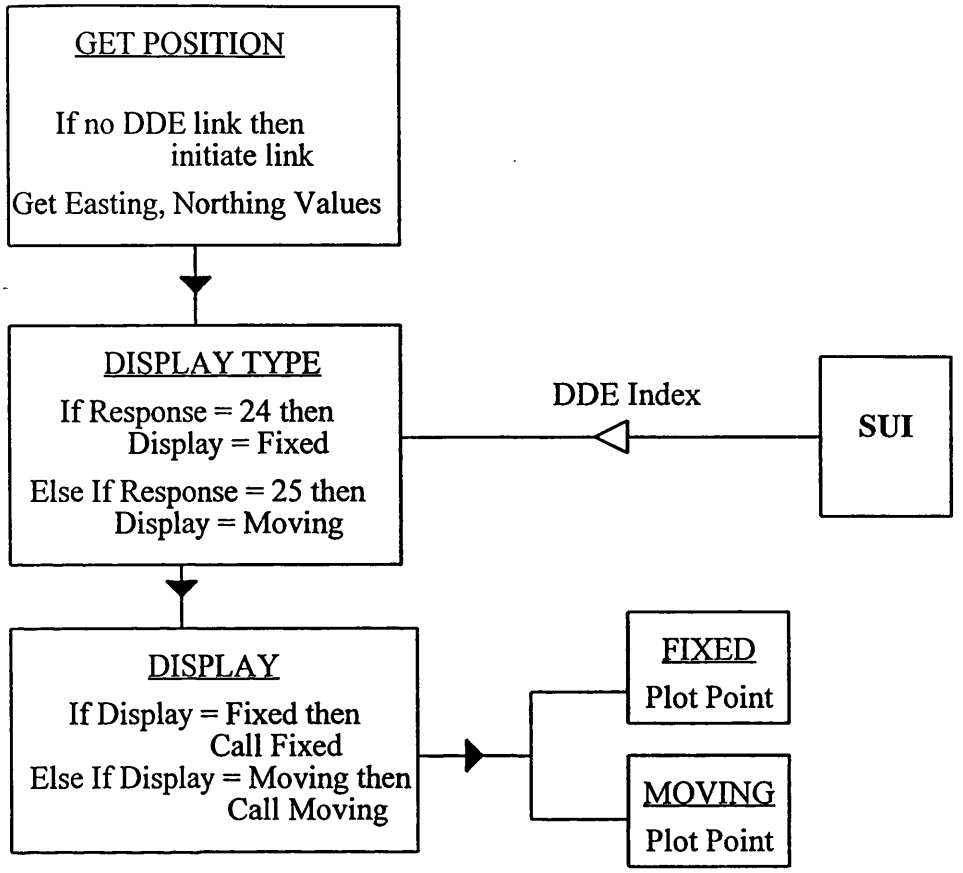


Figure 6.14 Block 2 to Block 1 Dynamic Data Exchange Link

(i) Get Position

The 'Get Position' procedure executed a two stage process:

(1) Check to see if a DDE link exists between Mapbasic and Visual Basic. If not then one is created.

(2) Get Easting and Northing values and assign them to variables.

Stage (1) was coded out using an 'If...Then' statement which operated as follows:

```
If no DDE Link exists Then
        Initiate DDE link with Visual Basic Group
End If
```

Stage (2) obtained the co-ordinate information using the DDERequest$ command. This "pulled" through the co-ordinate information and assigned it to two variables, East and North:

```
East = DDERequest$(i_chan,"txtEast")
North = DDERequest$(i_chan,"txtNorth")
```

(ii) Display Type

Once the information had been stored in variables, it could then be plotted and stored. Two methods could have been used to perform plotting, a fixed or moving map solution. The method employed was to be specified by the user when the GPS Driver was initiated via the SUI. Therefore once the 'GetPosition' procedure had been called, a 'DisplayType' procedure was called to check which solution the user has chosen:

```
If Response = "24" Then
        Display = Fixed
ElseIf Response = "25" Then
        Display = Moving
End If
```

This routine checked the DDE index obtained when the user chose the display type. If the index = 24, then a fixed display was selected and the variable 'Display' was given the value 'fixed'. If the index = 25 then display was set to 'moving'. This then stored the fix type selected by the user, and left the DDE index open for other instructions.

(iii) Display

Once the type of solution selected had been established, the relevant procedure could then be called, once again using an If...Then statement:

```
If Display = Fixed Then Call Fixed
Else
If Display = Moving Then Call Moving
End If
```

At this stage, the program recognised which solution to execute when plotting GPS information. The next stage was to actually plot the information and store it in the database.

The Statement Testing method, outlined by Abbot (1986), was used to test the GPS driver DDE link. A series of dialogue boxes were employed to ensure each statement was operational during run-time. The test was arranged so that the GPS receiver was set up over a stationary point in the field. The receiver was then switched on using the SUI and once a fix was established the Electronic Navigation Environment attempted to plot a point. Dialogue boxes were used to ensure the following steps were operational:

Index integer "99" sent to MapBasic group:

"Initiation OK"
"Link Index 99"

To establish that the MapBasic-Visual Basic link was operational:

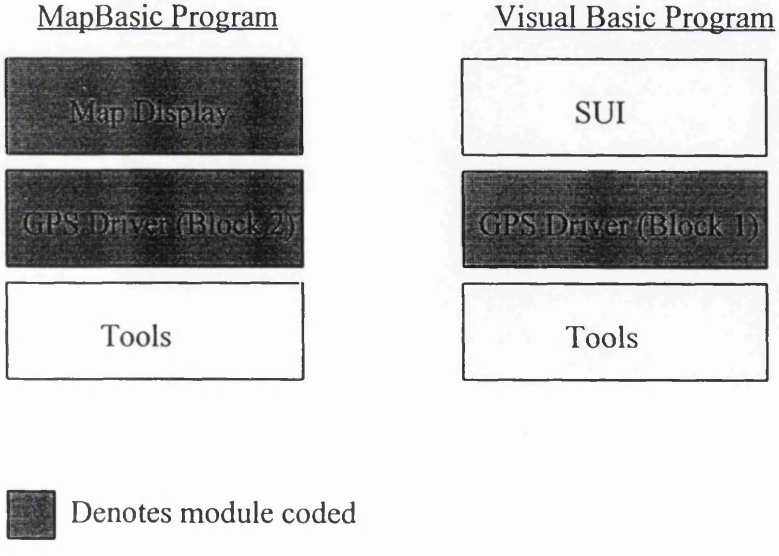
"GPS link established"
"Easting" (followed by Easting value)
"Northing" (followed by Northing value)

Once point had been plotted on the map:

"Point plotted OK"

Once the final message had been displayed it meant that the process had been completed successfully and each statement had been successfully executed. This was achieved on the initial test.

When the two Blocks in the GPS driver interacted successfully, Block 2 was integrated with the Map Display module. Block 1 formed the basis of the Visual Basic Program Group:



6.4 Tools

The development and coding of the tools was carried out in MapBasic, with the exception of the Time Estimation tool, which was developed in VisualBasic. Many of the tools could be created using MapInfo commands, making coding simple. Other tools had to be specially coded to achieve their aims. The Tools were coded and tested as independent modules using the methods described by Brown and Sampson (1973). The tools that could be created using MapInfo commands were as follows (Table 6.1):

Function	Procedure	Mapinfo Command
Open Map	OpenMap	M_FILE_OPEN
Save Map	SaveMap	M_FILE_SAVE
Close All	CloseAll	M_FILE_CLOSE_ALL
Register Map	Register	M_TABLE_RASTER_REG
Adjust Image Styles	Adjust	M_TABLE_RASTER_STYLE
Select All Routes/WP	Select	M_TOOLS_SEARCH_RECTANGLE
Delete Route/WP	Delete	M_EDIT_CLEAR
Pointer	Cursor	M_TOOLS_SELECTOR
Expand	ZoomIn	M_TOOLS_EXPAND
Shrink	ZoomOut	M_TOOLS_SHRINK
Ruler	Ruler	M_TOOLS_RULER

Table 6.1 MapInfo Commands as Electronic Navigation Tools

Each of these tools were created from a single MapInfo Command. The function column shows the definition of the tools task or function. The procedure column shows the name given to the function in the navigation environment. Finally the MapInfo column shows the command word required to invoke the function. All of these tools were coded in the same manner:

Sub *Procedure Name*
    Run Menu Command *COMMAND_NAME*
End Sub

This was all the code required to create the above tools. The procedure name was replaced with the corresponding name from the procedure column, and the command name with the corresponding MapInfo command. For example to code the register map function:

```
Sub Register
    Run Menu Command TABLE_RASTER_REG
End Sub
```

The majority of tools were coded in this way. However, there were a number which required more complex coding:

- Plot Route
- Plot Waypoint
- Recentre
- Time Estimation

The plot route and waypoint tools were coded in the same way as the tools mentioned above. However an extra piece of code was required to set the line and point style. This was achieved using the 'Set Style' Command. A sub command then specified line or point using 'Symbol' and 'Pen'. Finally a series of variables were entered stating the style of the object, colour and size. These procedures were therefore coded as follows:

```
Sub Plot_WP
        Set Style
            Symbol MakeSymbol(34, YELLOW, 18)
        Run Menu Command M_TOOLS_POINT
End Sub

Sub PlotRoute
        Set Style
            Pen MakePen (2, 2, magenta)
        Run Menu Command M_TOOLS_POLYLINE
End Sub
```

The recentre function required more complex coding. It employed the same code as used in the moving procedure from the GPS Driver. However, when testing this procedure, it was discovered that if no co-ordinate values existed to centre the map it was centred on (0,0). Therefore a query was set up asking the user to confirm this function:

```
Dim centrefix As Logical
'Open Table "POSITION"  Interactive

If East = 0 and North = 0 then
  centrefix = Ask("The co-ordinate values you will recentre on are (0,0), are you sure you want
to do this?", "Yes","Cancel")

If centrefix = False then
        Exit Sub
  End If
Else
End If

Insert Into POSITION
  Values(East,North)

Fetch First from POSITION
  Set Map
        Centre (East,North)
        CoordSys NonEarth
        Units "m"
  Create Point
        (East,North)
Symbol MakeSymbol(33, Magenta, 12)
```

Finally, the Time estimation tool was coded. This was created in Visual Basic as it operated independently of the Map Display and MapInfo. Values for speed and distance were entered by the user via list boxes in the SUI. Changing the value in one list box triggered the time calculation which was then output in a test box:

```
Dim dist As Variant
Dim speed As Variant
Dim time As Single
Dim Hour As Integer
Dim Minute As Integer

dist = cboDist.Text
speed = cboSpeed.Text
time = dist / speed
Hour = Fix(time)
Minute = (time - Hour) * 60
txtHour.Text = Hour
txtMinute.Text = Minute
```

This code was entered in the CLICK procedure of the distance and speed list boxes, so that change in either would invoke the code (Figure 6.15).

Clicking on either Box activates the 'CLICK' procedure which contains the time calculation. The resultant output is sent to the Journey Time Text Box for display.

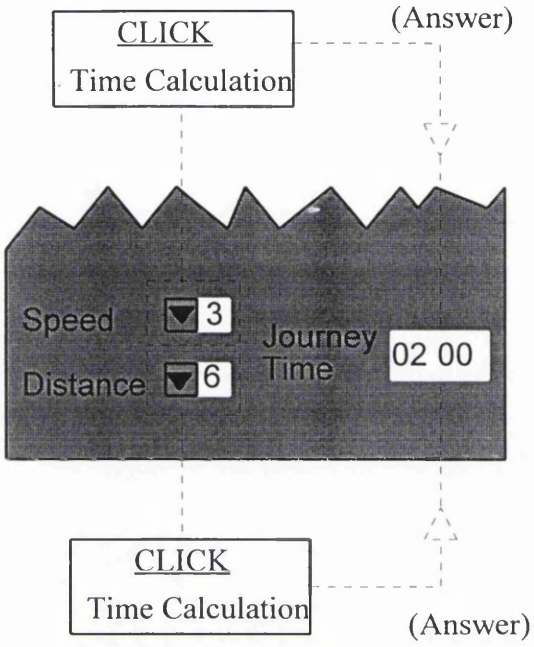
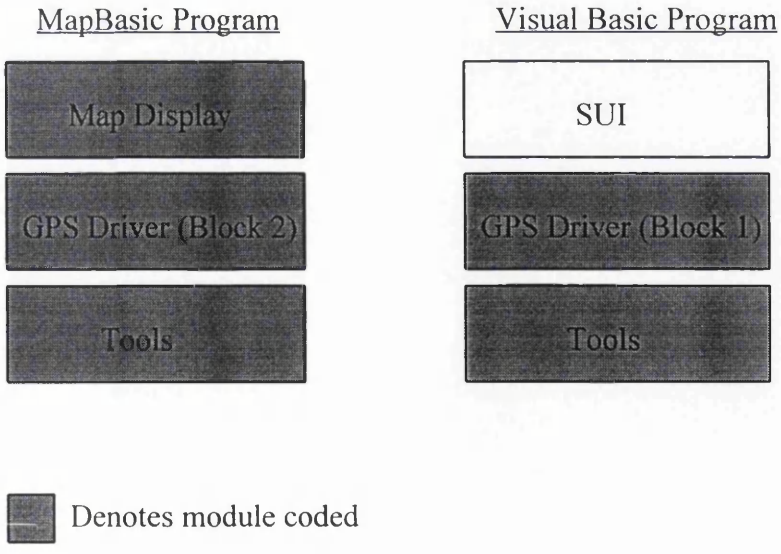


Figure 6.15 Time Estimation Code Location

With the Tools coded, the following Electronic Navigation Environment modules were completed:



6.5 Software User Interface Development

The Software User Interface (SUI) was coded using Visual Basic. Visual Basic programming requires two processes:

- Visual Programming
- Code Writing

6.5.1 Visual Programming

The Visual Programming section involved creating the visual aspects of the SUI and determining how it would appear to the user. Code writing was used to add functions and procedures to the visuals created. Each visual was created and then coded in accordance with the method laid down by Gurewich and Gurewich (1995).

The visuals were created using the design specification established for the SUI. This design was adhered to as closely as possible; however, in some cases Visual Basic was not capable of creating the required tools or graphics and so alternative designs had to be created.

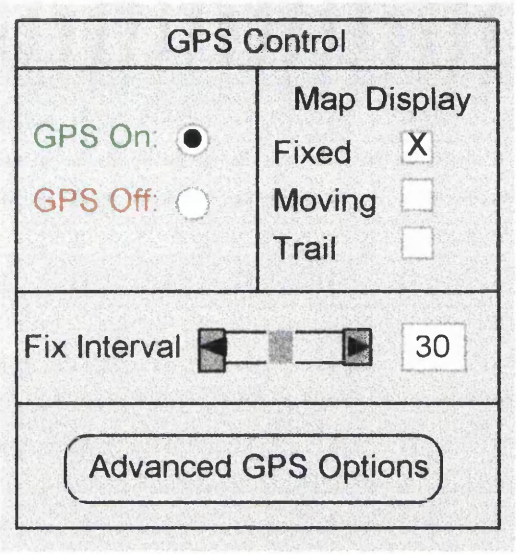
This first occurred while creating the file menu and the Exit command. The colour of the text on the command button was originally designed to be red. However, the command button text colour could not be changed from black, and it was decided to give the exit button a red border instead, as a warning. This redesign had a greater visual impact than the original idea and was seen as an improvement on the original design (Figure 6.16).

Original Design

Revised Design, with greater visual impact

Figure 6.16 Exit Button Redesign

Redesigns were also required in the GPS menu. When created to the original design specification the menu appeared visually unattractive (Figure 5.37), comprising mostly of a series of command buttons. It was also realised that the fix interval represented by a list box would be impractical. It was therefore decided to redesign the display options as check boxes and the fix interval tool as a scroll bar. This would take up less room than a list box and these redesigns gave the menu a more appealing look (Figure 6.17).



6.17 GPS Redesign

It should also be noted from Figure 6.17 that the GPS on/off also had to be redesigned. The radio buttons could not be coloured as in the original design so the text that accompanied the buttons was coloured instead.

The final set of redesigns were required on the Route planning menu. In the original design, the command buttons were to be annotated with graphics (Figure 5.40). However this was not possible in Visual Basic and so graphical icons were created instead. These were then given Text labels (Figure 6.18).

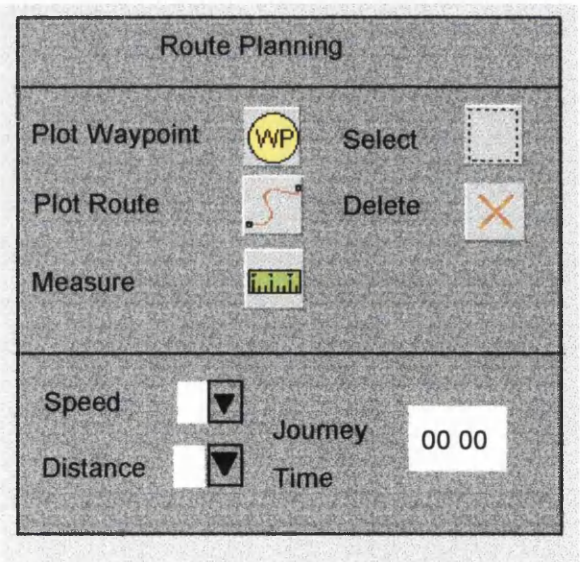


Figure 6.18 Route Planning Redesign

Each menu was created as a separate form in Visual Basic. Once the visual programming was completed and each form had been drawn, the next stage was to code the forms.

6.5.2 Coding

The coding of the SUI involved icon animation and form linking. The SUI was coded using the Top-Down methods laid down by Yourdon (1975) and McGowan and Kelly (1975).

(i) Icon Animation

The icons in the SUI were animated so that when a selection was made by the user, they could physically see the selection (Kobara, 1991). This involved making the icon appear to depress as if it were a button being pressed. In order to carry this out two copies of each icon were made. The first icon was made with a dark grey background and 3D shading. The second icon was made with a lighter background and reversed 3D shading (Figure 6.19). If an icon was selected, it changed from one shading to the other and appeared to be depressed.

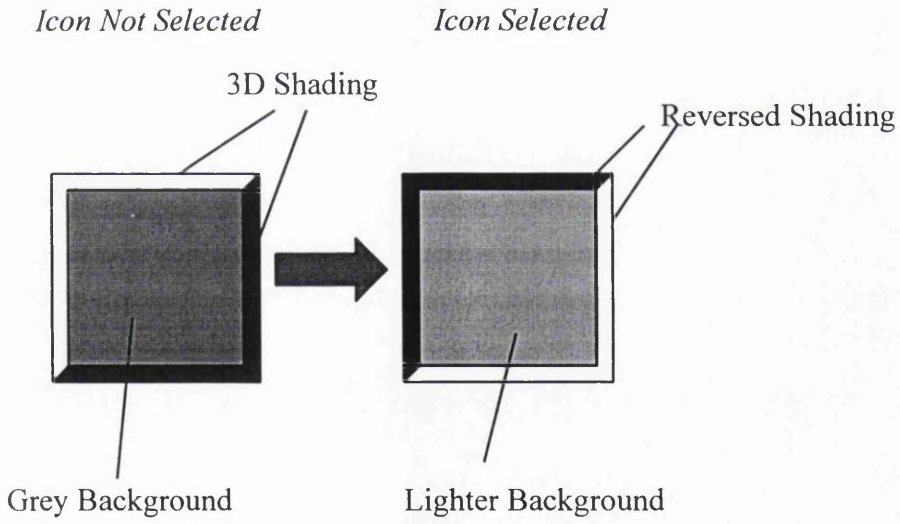


Figure 6.19 Icon Design for animation

In order to animate these images, the command 'Load Picture' was used. The original darker icon was shown by default. If the icon was selected the second image would be loaded into the image space making the icon appear depressed:

```
pictZoomin.Picture = LoadPicture("c:\Psp\zoomin2.bmp")
pictZoomin.Picture = LoadPicture("c:\Psp\zoomin.bmp")
```

The original image would then be loaded in once the icon was deselected.

(ii) Form Linking

This coding involved the linking of the relevant menus to each other, so that if the GPS icon on the toolbar was selected, the GPS menu would appear. This was achieved by using the 'Show' command, which was preceded with the relevant form's name. For exapmle, to link the GPS icon and menu the following code was inserted into the CLICK procedure of the GPS icon:

```
frmGPS.Show
```

This would then display the GPS pop-up menu (named frmGPS) (Figure 6.20).

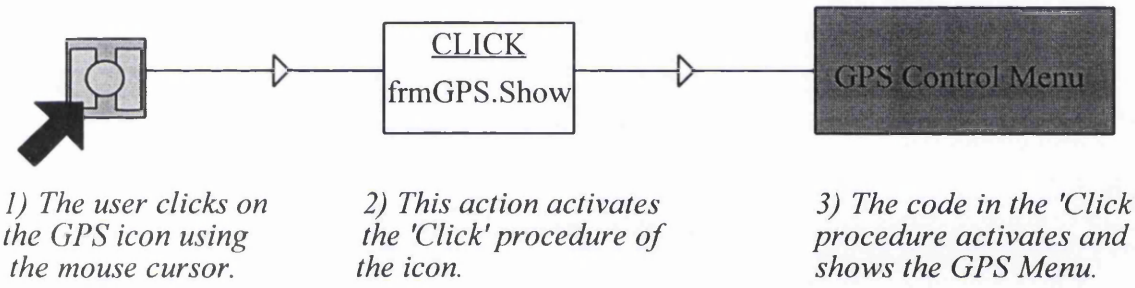


Figure 6.20 Form Linking

The form linking created the navigation environment menu system, and to ensure the correct menus were selected a branch testing technique was adopted as described by Abbot (1986). This essentially involved selecting each menu item in the SUI to ensure that every form was linked correctly within the menu system. The branch testing revealed that the forms in the SUI had been linked correctly.

6.6 Visual Basic to MapBasic Group Dynamic Data Exchange Index

The final stage in developing the Electronic Navigation Environment was to establish the DDE link between the Visual Basic and MapBasic groups. As discussed at the design stage, an indexing system was used. Each tool and function was given a unique identifying number or index (Table 6.2). These were grouped according to function type. Gaps were left in the index in case new tools or functions had to be created.

Tool Bar	Index	**Route Planning**	Index
Zoom in	1	Plot Route	31
Zoom out	2	Plot Waypoint	32
Recentre	3	Select	33
Cursor	4	Delete	34
Map Move	7	Measure	6
GPS	Index	**File Functions**	Index
GPS Off	22	Open File	11
Moving Map Display	25	Save File	12
Fixed Map Display	24	Close File	13
Trail	27	Adjust	14
GPS Fix	99	Register	15
		Exit	16

Table 6.2 Index Integers

6.6.1 Dynamic Data Exchange - Software User Interface Link Coding

The Dynamic Data Exchange - Software User Interface coding was undertaken using the top-down method, as described by McGowan and Kelly (1975). The DDE process required all involved applications to be running simultaneously during conversation so these had to be launched before any conversation could take place (Figure 6.21).

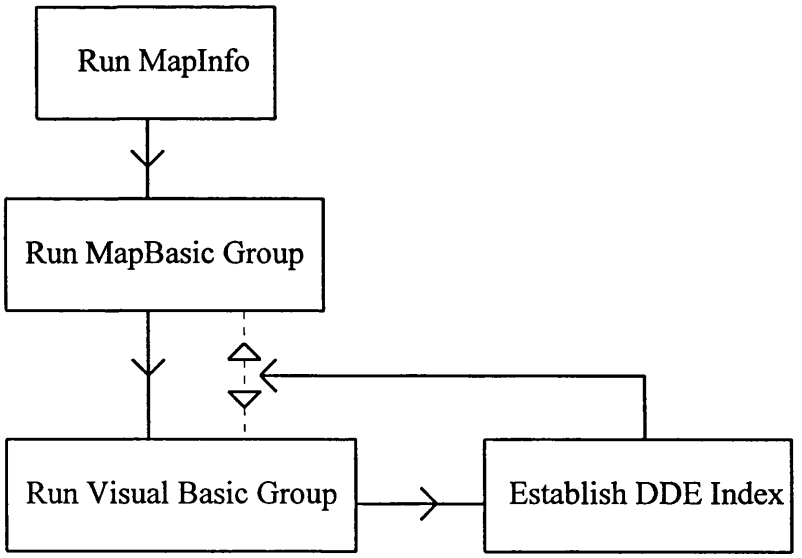


Figure 6.21 Application Running

Establishing the link between Visual Basic and MapBasic functions was a fairly straightforward task, made even simpler through the use of the indexing system. The link was made using the following process:

```
txtResponse.LinkTopic = "MapInfo|vbdata"
txtResponse.LinkItem = Response
txtResponse.LinkMode = 2
```

where:

txtResponse = Index Integer Holding Cell

Response = Index Integer

Link Topic - Specifies the server program and the system it runs on e.g.

MapInfo = MapBasic Program Grouping's System

vbdata = Name of the executable file that contains the MapBasic Program Group

LinkItem - Specifies the variable or item to be involved in the DDE conversation.

LinkMode - Specifies type of link to be Initiated i.e. Automatic (2) or Manual (1). An Automatic link means that whenever the Response integer changes the link is updated.

This code initiated the DDE link. The relevant index integer could then be sent:

```
txtResponse.LinkExecute txtResponse.text
```

where:

LinkExecute - Sends the information across the DDE link

txtResponse.text - Specifies the information to be sent.

When the index integer was received by the MapBasic program group it entered the Remote Message Handler. Here the index entered a set of conditions to ensure the correct procedure was called:

```
Sub RemoteMsgHandler
    If Response = "1" Then
        Call Zoomin
    ElseIf Response = "2" Then
        Call Zoomout
                :
                :
                :
    ElseIf Response = "34" Then
        Call Delete
    ElseIf Response = "99" Then
        Call GetPosition
        Call Display
    End If
End Sub
```

6.6.2 Integration Testing

Testing was carried out at this stage to establish the DDE index was functioning properly. The Statement Testing method as described by Abbot (1986) was employed. The testing process was carried out in two stages:

(i) Ensure index system was operational
(ii) Ensure Main DDE link was correctly established

(i) Index System

In order to test the index system, a text box was set up and displayed on screen whilst the Software User Interface was running. When a function was selected the index integer was displayed in the text box and compared with the index listings (Table 6.2) to ensure the correct integer corresponded to the right function. The results of this test are given in Table 6.3.

Tool Bar		Route Planning	
Zoom in	1	Plot Route	31
Zoom out	2	Plot Waypoint	32
Recentre	3	Select	33
Cursor	4	Delete	34
Map Move	7	Measure	6
GPS		**File Functions**	
GPS Off	22	Open File	11
Moving Map Display	25	Save File	12
Fixed Map Display	24	Close File	13
Trail	27	Adjust	14
GPS Fix	99	Register	15
		Exit	16

Table 6.3 Index Test

This simple test proved that the statements executed successfully and therefore index system was operational.

(ii) Dynamic Data Exchange Link Test

Tests were carried out on two levels to ensure the main DDE link was operational. The first involved a series of messages being printed in dialogue boxes in MapInfo once the link had been initiated. If no link was established then a different series of dialogue boxes would have been displayed. For correct initiation the following messages were displayed:

"Initiation OK"
"Link Index" (followed by the index integer selected)

For incorrect initiation:

"Initiation unsuccessful"

This was seen as the ideal method for testing the establishment of the DDE link and once the first set of dialogue boxes were displayed for each index integer, the link was deemed operational.

The second level of tests required that not only the link be established, but the relevant function be performed by the system. For this purpose a map sheet was set up in MapInfo and the various functions were performed on the sheet (such as zoom in, zoom out, plot

point etc). Again a series of dialogue boxes were employed to ensure step by step operation. For example, if the Zoom in icon was selected in the SUI:

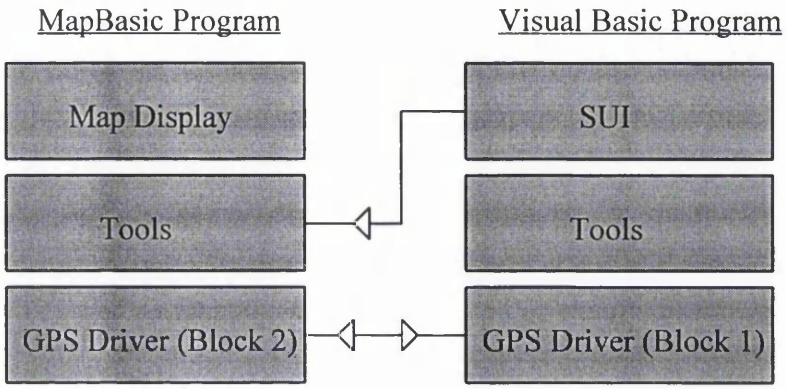
"Initiation OK"
"Link Index 1"
"Zoom in procedure activated"

The zoom in tool was then tested on the map sheet to ensure that it operated correctly. This process was repeated for each function. Each proved to operate correctly once the first stage of tests had been completed and so the DDE system was deemed operational.

The success of these simple tests illustrated that the DDE links in the Electronic Navigation Environment were fully operational. The display of dialogue boxes at each stage in the testing procedures ensured that if an error occurred it was located easily in relation to where the dialogue box code was situated within the program listing.

6.7 Summary

This concludes the development. Each module had been coded and tested, then integrated into the Electronic Navigation Environment. DDE links had also been established between the Software User Interface and MapBasic Tools, and Block 1 and Block 2 of the GPS Driver.



Although on the whole the development of the Electronic Navigation Environment closely followed the original design, changes had to be made. All of these changes were created by either problems with programming logic (the GPS DDE link), real-time problems (Fix Interval) or language limitations (Software User Interface redesigns). The changes that were made to the original design often improved on it. This was especially true of the Visual Basic Timer used instead of a fix interval loop.
This highlighted the problem between the designer's visualisation of what he/she wants with what can be produced with the tools provided. These problems also illustrated that a comprehensive knowledge of all development tools should be obtained before design. However, this is not usually possible, especially when several development tools are being used. The designer cannot be expected to be expert in all systems instantly, and so redesign and redevelopment are inevitable.

The completion of the navigation environment did not signal the end of development and design. The next chapter discusses and explains how the Electronic Navigation Environment was assessed, and what redesigns or new developments arose from this assessment.

CHAPTER 7: TESTING, RESULTS & ANALYSIS

7.1 Electronic Navigation Environment Testing

This chapter covers the system testing of the Electronic Navigation Environment, the results gained from the tests and how these results reflected on the navigation environment in terms of the user requirements. The programs and modules that comprised the navigation software had been tested during development (see Chapter 6) to ensure that they operated correctly. However, as stated by King (1988) this was no guarantee that when implemented as a single system, these modules and programs would still operate as required. Four types of test were carried out on different aspects of the Electronic Navigation Environment to determine its functionality, usability, and operation. The tests carried out were:

- Map Display Testing
- GPS Driver Testing
- Tools Testing
- Software User Interface (SUI) Testing

7.2 Map Display Testing

The map display was tested to ensure that it would operate as required with both raster and vector data. It was already known that MapInfo could display both types of data; however, it had to be established whether the Electronic Navigation Environment system as a whole had any effect on the map display and vice versa.

7.2.1 Raster Data Display

Problems arose when the user tried to interact with the navigation environment after a raster map had been loaded into the map display. If the user then tried to select a tool from the SUI, the map display would flash and, in some cases, the colours on the map display would be corrupted (i.e. the whole image could be tinted blue or green). There were thought to be several possibilities as to why these problems arose:

(i) Not enough memory to process the change between windows efficiently.
It was thought initially that the 'flashing effect' was caused by the sheer size of the raster map image, causing the switch between the Map Display window and the SUI window to

slow down so much as to become visible. However, tests were carried out with smaller raster images and the effect still occurred (Table 7.1).

(ii) Mis-Match in colours between the two windows.

This suggestion was based on the assumption that each window was operating on a different colour scale (i.e. the SUI icons were displayed in 256 colours while the Map Display depended on the data shown, usually 16 or 256 colours). To test this theory, a raster image was altered and set at 256 colours, the same as the SUI. This image was then loaded into the navigation environment, but the flashing and tinting, although less pronounced, still occurred.

The cause of the flashing and tinting seemed to be caused by the use of two different colour palettes, and the colours in the map display not matching those in the SUI. If a black and white raster image was loaded, no flashing and tinting took place, regardless of the image size.

The solution to the problem would be to exactly match both sets of colours. However, because the colour depth varies from one set of raster data to another this would be impractical. The alternative would be to fully integrate the SUI with MapInfo, so that only one colour palette is used by both map display and SUI. This solution would be outwith the scope of the current project, but could be a major future development.

Image Type	**Flashing / Tinting?**
4Mb Raster Image (256colours)	Yes
4Mb Raster Image (64colours)	Yes
4Mb Raster Image (16colours)	Yes
0.17Mb Raster Image (256colours)	Yes
0.17Mb Raster Image (64colours)	Yes
0.17Mb Raster Image (16colours)	Yes
0.5Mb Raster Image (2colours (B&W))	No

Table 7.1 Raster Data Testing

7.2.2 Vector Data Display

The Map Display was capable of supporting vector data. Interactions with the rest of the navigation environment were not affected in any way when vector data was displayed and similarly, the Map Display showing vector data was not adversely affected by the remaining

Electronic Navigation Environment functions. The Map Display was fully functional when using vector data.

7.3 GPS Driver Testing

The following aspects of the GPS Driver were tested:

- Transformation Accuracy and Consistency
- Graphical Plotting Accuracy
- Operation

7.3.1 Transformation Testing and Analysis

As stated in Chapter 5, a transformation was required to convert the co-ordinates given by the GPS Receiver in World Geodetic System 1984 (WGS84), to Ordnance Survey National Grid (OSNG), so they could be plotted on the digital maps of Great Britain. A test was carried out to ensure the transformation operated accurately and consistently.

(i) Test Data Set

In order to carry out the test, a data set of co-ordinates was acquired. The data set consisted of 27 points, with co-ordinate values in both WGS84 and OSNG (Table 7.2).

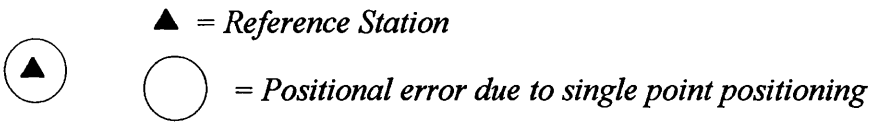
	WGS 84		OS NG	
Point	**Latitude**	**Longitude**	**Easting**	**Northing**
1	00 07 28.569428	51 24 10.298762	547878.412	169231.268
2	00 08 07.621447	51 36 44.144403	547953.500	192538.343
:	:	:	:	:
:	:	:	:	:
27	01 05 02.573045	51 06 16.537812	615996.833	138442.611

WGS84 co-ordinates are shown in degrees, minutes and seconds
OSNG co-ordinates are shown in metres

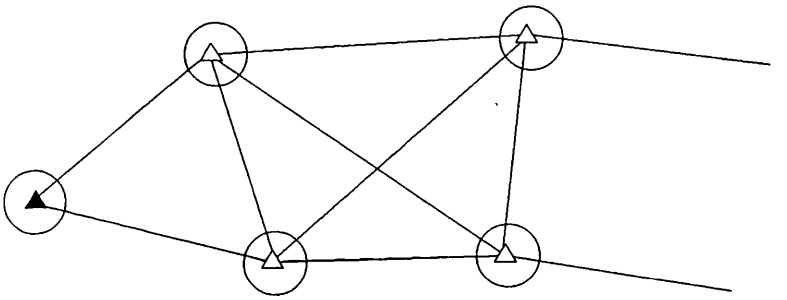
Table 7.2 Test Data Set

The data set points were originally used to provide control for a major construction project and were derived using Leica GPS receivers and SKI processing software. The WGS84 co-ordinates of Points 2-27 were based on the WGS84 co-ordinates determined for Point 1 (known as a reference station). The WGS84 co-ordinates for Point 1 were determined using single point positioning techniques which incurs random errors in the point due to Selective

Availability (Gilbert, 1995). The required accuracy of Point 1 is +/- 10 metres (Leica, 1993) which, according to Leica (1994) should have been achieved by collecting observations for a single point over two to three hours and averaging the results. This meant that the WGS84 co-ordinates in the data set were not absolute, and any error that existed at Point 1 would have also occured at the points determined relative to it (i.e. Points 2-27) (Figure 7.1). In addition to the error that existed throughout the points due to single point positioning, there would also have been random errors at each of the points.



a) The co-ordinates of the reference station were established using single point positioning to an accuracy of approximately +/- 10 metres.



b) Because all other points in the network were determined in relation to the reference station, the error present in the reference station would also be present in all points of the network.

Figure 7.1 Errors due to Points Based on a Single Reference Point

(ii) Method

A direct comparison could not be made between the given OSNG values and the transformed OSNG values, because of the errors that could exist in the given WGS84 co-ordinate values due to their method of derivation. Therefore, a test was established to compare co-ordinate differences between the co-ordinates of the given points and the transformed points.

The WGS84 values from the test data set were transformed to OSNG values using the transformation code. Once the transformed values had been obtained, the differences between the co-ordinates were calculated (Table 7.3). This was carried out for 364 different combinations of points. This process was then carried out for the given OSNG values in the test data set. This gave a total of 364 pairs of values to compare.

		Easting	Northing	dE	dN	
Transformed Values	1	547881	169234	-75	-23306	Difference in:
	2	547956	192540			dE / dN
						0.088 / 1.075
Given Values	1	547878.412	169231.268	-75.088	-23307.075	
	2	547953.5	192538.343			

Table 7.3 Example of Comparison of Co-ordinate Differences

The differences were then compared in Eastings, Northings and difference in distance. The standard deviation, mean, maximum and minimum values were calculated, using the equations and guidelines discussed by Hammond and McCullagh (1989) and Mikhail and Gracie (1981), to establish whether or not the differences fell within the specified accuracy of 2 metres. The test data points were then plotted with the transformed points using the OSNG co-ordinates, and the direction of shift between the values was calculated to establish whether the transformed co-ordinates had shifted uniformly or randomly. In summary, the test involved the following steps:

- Transform the test data set of WGS84 co-ordinates to OSNG using code.
- Calculating the difference between all combinations of the test data OSNG co-ordinate values.
- Calculating the difference between all combinations of the transformed OSNG co-ordinate values.
- Compare the co-ordinate differences between the test data set and the transformed data.
- Calculate direction of shift from test data OSNG position to transformed OSNG position.

(iii) Results

Table 7.4 shows the results of the analysis carried out on the differences between points in the given test data and the transformed data.

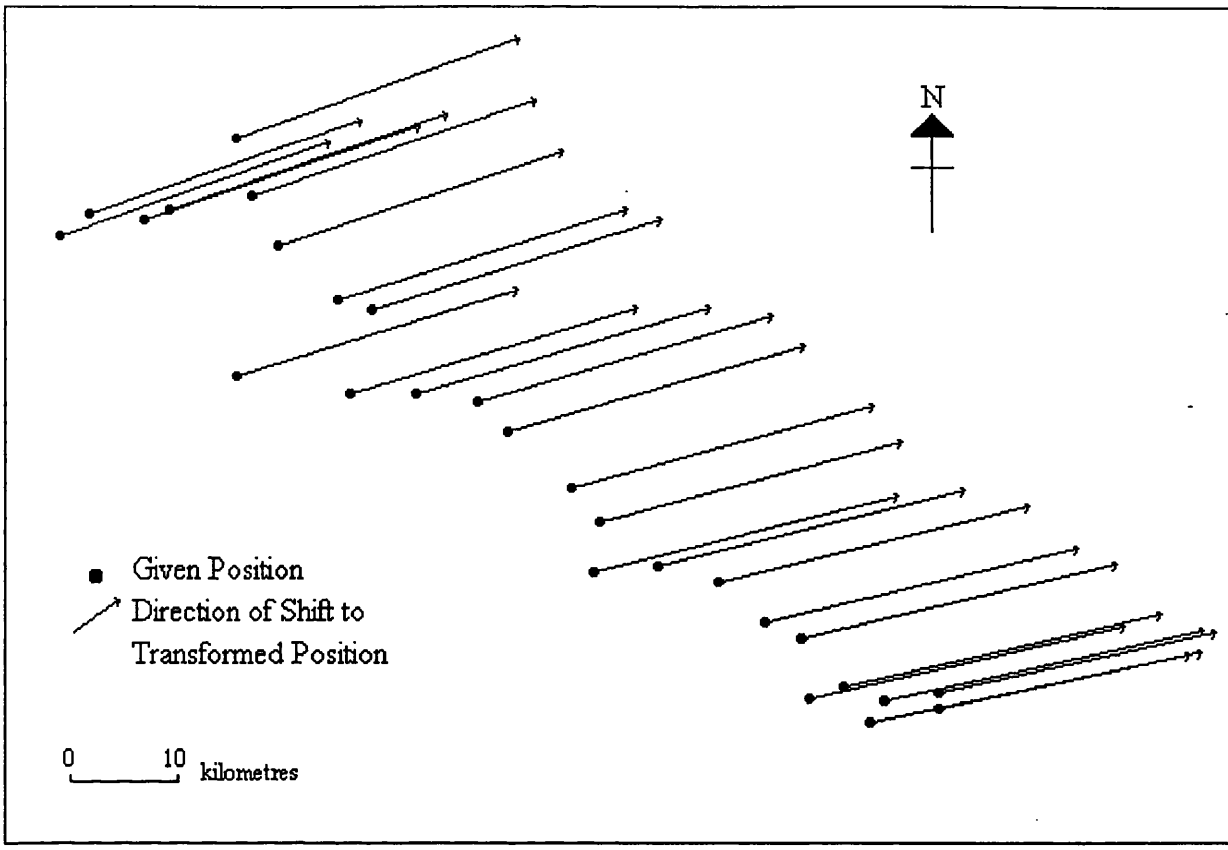
	Difference in:		
	dE	dN	Distances
Standard Deviation	0.253	0.263	0.275
Mean	0.365	0.379	0.578
Maximum	1.061	1.251	1.305
Minimum	0.009	0.001	0.038
Population Size	364		

All figures are in metres

Table 7.4 Transformation Accuracy Test Results

As can be seen from Table 7.4, the standard deviation of the differences between the test data set point co-ordinate differences and the transformed point co-ordinate differences was consistent for Eastings and Northings with only a 1cm difference between them. The mean distance between the given point differences and the transformed point differences was found to be 0.578 metres, with a maximum vector error of 1.305 metres and a minimum of 0.038 metres. This illustrates that all the values compared fell within the specified accuracy of the transformation, which was 2 metres.

The test data set points and the transformed points were then plotted, to investigate whether the direction of the difference between the co-ordinates was uniform for all points or random (Figure 7.2)



Magnitude of shifts is exaggerated for clarity

Figure 7.2 Direction of Difference between True Co-ordinates and Transformed Co-ordinates

Plotting the test data and transformed points appeared to show a uniform direction in the shift between the given and transformed co-ordinates (Figure 7.2). However, inspecting the individual direction between test data points and the corresponding transformed points revealed that the direction varied from 33.536417 degress to 54.913631 degrees (Table 7.5). This suggested that the transformed points were shifted generally in a north east direction, but random errors ensured that the shift was not truly uniform.

Point	Direction of Shift (degrees)	Point	Direction of Shift (degrees)	Point	Direction of Shift (degrees)
1	46.550485	**10**	37.726641	**19**	46.008473
2	33.536417	**11**	54.913631	**20**	43.875648
3	43.672580	**12**	44.606221	**21**	43.931329
4	48.461308	**13**	52.102067	**22**	53.291529
5	40.766314	**14**	36.608259	**23**	45.961517
6	46.555853	**15**	41.641330	**24**	47.341444
7	42.525869	**16**	40.189227	**25**	42.831563
8	49.018161	**17**	52.130859	**26**	44.075623
9	45.000000	**18**	41.648359	**27**	47.789642

Table 7.5 Direction of Shift from Given Points to Transformed Points

The appearance of a general uniform shift could be attributed to either the test data or the transformation. As already explained, the test data WGS84 co-ordinates were based on a single point position and so any errors in this point would also exist within the rest of the WGS84 co-ordinates. When these co-ordinates were then transformed, these errors would cause a uniform direction of shift. However, the existence of random errors at each of the points would ensure that this direction was not truly uniform, and would vary at each point depending on the magnitude of the random errors at that point.
Alternatively, if there was no error in the single point position on which these co-ordinates were based, then the general shift could be attributed to the transformation, with the variations in the shift direction being attributed to the random errors that existed at each point. Unfortunately it was not possible to determine the cause of this shift, although it is highly probable that there was in fact a difference between the true and single point positions of the reference station, with the shift in co-ordinates coming primarily from this source.

What can be stated from the test and results is that the transformation functioned satisfactorily and within the tolerances and specifications outlined in Section 5.4.1.2.

7.3.2 Plotting Accuracy and Operation Tests

The *accuracy* of the GPS Driver related mainly to ensuring that co-ordinates were plotted in the correct position on the map. This process had to be repeated over a number of days to ensure that the GPS Driver would provide consistent results. The *operation* of the GPS Driver related to the ability of the driver to plot co-ordinates in real time, both whilst the

receiver was static and whilst mobile. To this end two tests were devised for the GPS Driver:

- Static Test
- Mobile Test

(i) Static Test

The static test was used to evaluate the accuracy of the GPS Driver. This involved setting up the GPS receiver over a single point, with known Ordnance Survey National Grid (OSNG) co-ordinates. The Electronic Navigation Environment was executed on a laptop computer fitted with the GPS personal computer card. The receiver was then initiated and positional data collected and plotted on-screen. To collect the data, it was recorded using the Trimble GPS Control Panel at a sample rate of one fix per second (this rate was chosen so as to collect the maximum amount of data possible). The data was then analysed to ensure the GPS Driver met the specified in section accuracies. This test also examined the operation of the GPS Driver in static mode. If points were plotted on screen, then the driver was deemed operational. The test procedure was therefore as follows:

- Set up GPS Receiver over known point
- Execute Electronic Navigation Environment
- Set Trimble GPS Control Panel on Record Mode
- Initiate GPS receiver
- Record Data and inspect Map Display to ensure position plotting
- Stop data recording

The amount of data recorded depended on the battery power offered by the laptop computer. The battery in the computer typically lasted for between 10 to 30 minutes, depending on the amount of data stored on the hard disk drive. In extreme cases, where the battery only lasted 10 minutes, this gave very little time to record data once the experiment had been set up and a position fix established - typically 2 to 3 minutes.

The GPS receiver's antenna was placed on a tripod to ensure good ground clearance and improve reception, as reception was poor if the antenna was placed directly on the ground (Figure 7.3).

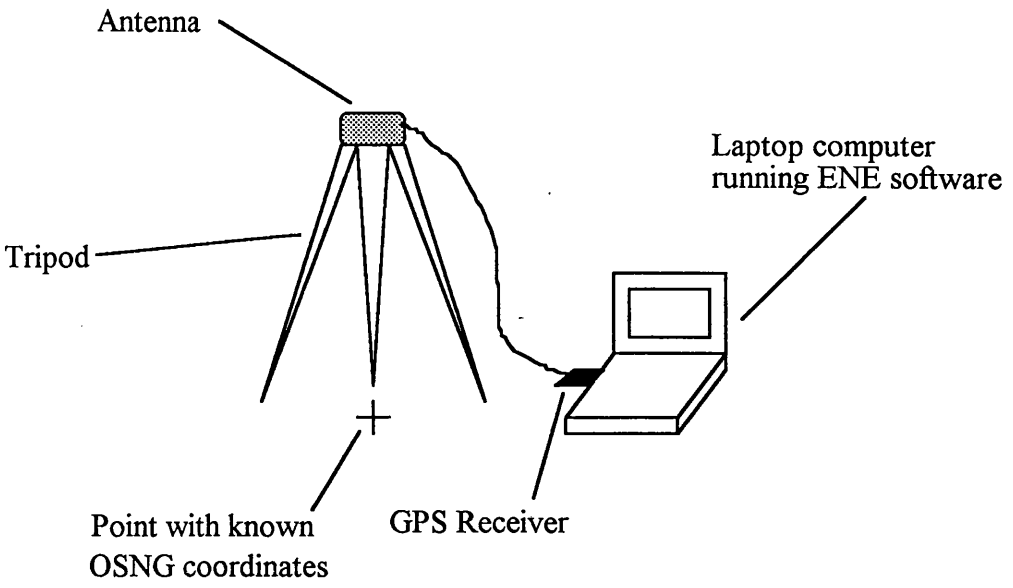


Figure 7.3 Equipment Set Up (Static Test)

(ii) Mobile Test

The Mobile test was designed primarily to evaluate the operation of the GPS Driver. As in the static test, the amount of data collected depended on battery power. Ideally the test would have been conducted on foot. However due to the limiting factor of battery power experienced during the static tests, a test on foot was not practical in terms of collecting a reasonable amount of data. It was therefore decided to conduct the test using a vehicle to maximise the amount of data that could be collected.

The GPS antenna was attached magnetically to the roof of the vehicle, with the receiver and portable computer running the navigation software located inside (Figure 7.4). The vehicle then followed a pre-determined route to ensure that positions were plotted in real-time, to within specified accuracies. This test was not as accurate as the static test, because the receiver was mobile and plotted positions could not be related to specific known points. However, accuracy was analysed as a check to ensure no gross errors occurred.

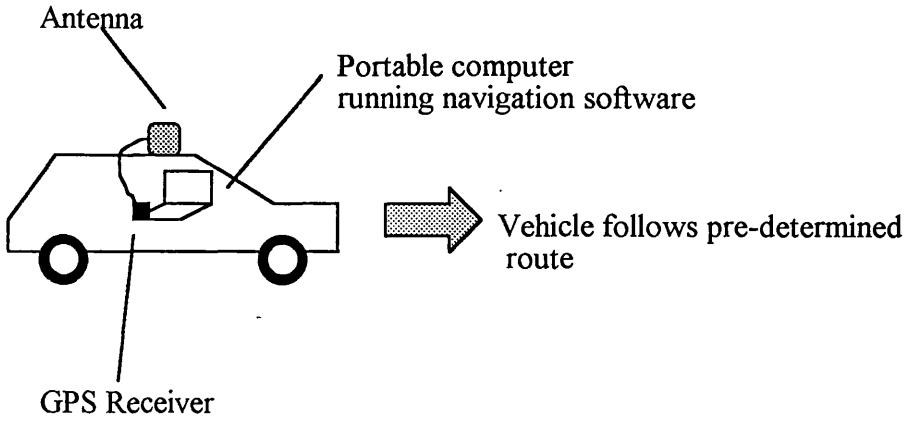


Figure 7.4 Equipment Set Up (Mobile Test)

Once a position fix was established, the car was driven round the pre-determined route with the data being recorded via the Trimble GPS Control Panel at a sample rate of one fix per second. Once the route had been completed, the data was played back and analysed to ensure operation. In summary the mobile test procedure involved:

- Attaching GPS antenna to vehicle roof
- Executing Electronic Navigation Environment
- Setting Trimble GPS Control Panel to Record Mode
- Initiating GPS receiver
- Recording data as car was driven round the pre-determined route
- Stopping data recording

The static and mobile tests were carried out over a number of days to ensure consistent results were achieved. A GPS Testing Sheet was filled out for each test, recording the date, time, weather conditions etc. (Appendix V).

(iii) Test Map Data

Prior to the static and mobile tests being carried out, suitable test map data had to be selected. Two sets of map data were used to test the GPS Driver:

- Ordnance Survey 1:1250 tile of City Centre, Nottingham
- Independent survey 1:2500 tile of Clifton Campus, The Nottingham Trent University

Both data sets were vector types. Vector data was chosen as it uses less memory than raster data, and therefore requires less battery power to manipulate. This increased the length of the testing time. The City Centre data was used for the static tests, as a station was located

there with known Ordnance Survey National Grid (OSNG) co-ordinates. The station was located on the roof of the Newton Building, The Nottingham Trent University, and so provided a convenient test site. The Clifton Campus data was used for mobile field tests. The site was chosen as it was University owned and therefore readily accessible. There was also little risk of the tests being interrupted by other traffic.

The map data was supplied in Drawing Exchange Format and had to be imported into MapInfo format. Once the maps had been imported, the co-ordinates of the map data were checked, to ensure the importing process had not adversely affected the map data.

In the case of the City Site data, the true values of the Ordnance Survey grid lines were printed on the digital map. Once the map data had been imported into MapInfo, the intersections of these gridlines were selected and the co-ordinate values that were subsequently displayed were compared with the printed values to ensure that the data had been imported successfully (Table 7.6). When the co-ordinates were checked there were no differences between the two sets of co-ordinate values and so the imported data was ready for use.

	Given Value		Imported Value			
Point	E	N	E	N	dE	dN
1	457000.00	340000.00	457000.00	340000.00	0	0
2	457500.00	340500.00	457500.00	340500.00	0	0
3	457500.00	340000.00	457500.00	340000.00	0	0
4	457000.00	340500.00	457000.00	340500.00	0	0

Table 7.6 City Site Co-ordinate Check

The Clifton Campus data was checked using co-ordinate values supplied by the Estates department at The Nottingham Trent University, which were scaled from an Ordnance Survey 1:1250 map sheet of Clifton Campus. Eight points were supplied, relating to common points such as building corners that appeared on both the digital map and the Ordnance Survey map sheet. The co-ordinates supplied for these eight points were specified as being accurate to +/- 1 metre.

The digital map was then imported into MapInfo and the common points selected to show the corresponding imported co-ordinate value. These co-ordinates were then compared to those scaled from the Ordnance Survey map sheet. The specified accuracy of the Clifton Campus data was +/- 20 metres. When co-ordinate values were checked on the imported map, they proved to be within this specification (Table 7.7).

Point	Given Value		Imported Value		dE	dN
	E	N	E	N		
1	454897	335254	454896	335253	1	1
2	454761	335272	454761	335272	0	0
3	454881	335374	454882	335362	-1	12
4	455089	335212	455088	335211	1	1
5	454863	335103	454859	335117	4	-14
6	454720	335085	454716	335102	4	-17
7	454965	335356	454966	335344	-1	12
8	454970	335184	454968	335188	2	-4

Table 7.7 Clifton Campus Co-ordinate Check

With the tests established and suitable map data selected, the testing was carried out and the following results were obtained.

7.3.4 Static Test Results and Analysis

Four static tests were carried out to assess the accuracy and static operation of the GPS Driver (Figure 7.5). The largest error that could be produced by the GPS Driver was 102 metres, as the accuracy of the co-ordinate transformation in the GPS Driver is 2 metres, and, according to Trimble Navigation (1994a) the GPS receiver is capable of calculating co-ordinates to +/- 100 metres 95% of the time.

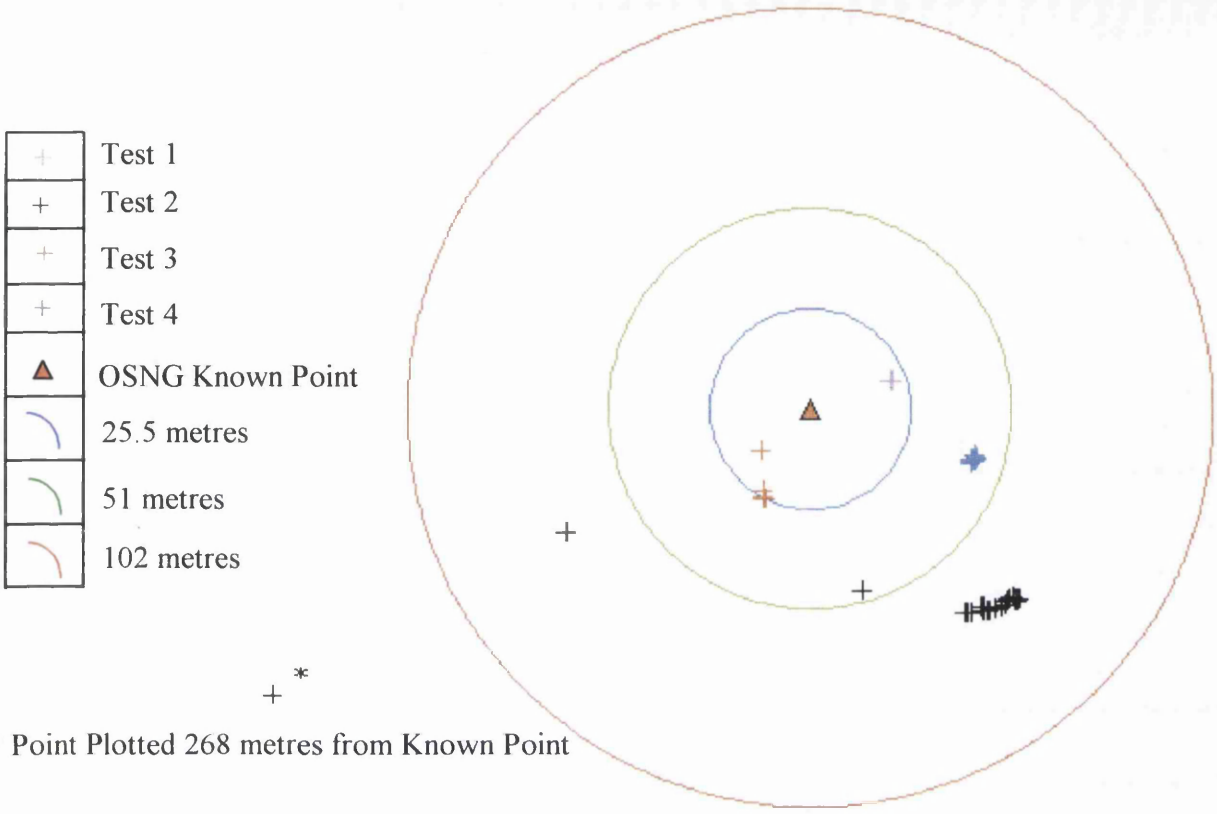


Figure 7.5 Static Test Results (Tests 1-4)

Of the test data collected, only 0.5% fell outside specified accuracy of 102 metres (Figure 7.6). This was caused by one point during Test 2 being plotted 268 metres from the OSNG known point. This point was the first to be plotted during Test 2 and successive points were plotted closer and closer together until the positions were plotting in one location. This was caused by the internal set up of the GPS receiver. Trimble Navigation (1994a) state that the receiver is set up to calculate a three-dimensional position which requires a minimum of 4 satellites. However, if only 3 satellites are available then the receiver switches to two-dimensional positioning mode. When the receiver obtains signals from the satellites they must pass a set of mask criteria that ensure the satellite signals will calculate a position that meets the accuracy requirements. One of the mask criteria is Position Dilution of Precision (PDOP). This is a unitless figure that, as discussed by Leick (1990) is used to describe the effect of satellite geometry on positional accuracy. Generally, the higher the PDOP is, the less accurate the position may be. In the case of Test 2, initial positions were gained in two-dimensional mode which requires a PDOP of 12 or less to calculate position. In contrast three-dimensional mode requires a PDOP of 5 or less in order to calculate position. Therefore, the initial positions that were calculated in two-dimensional mode were not as accurate as those plotted in three-dimensional mode and the positional error was high. As

the receiver switched to three-dimensional mode the mask criteria required a PDOP of 5 and so the required accuracy is increased, improving the accuracy of positions.
The majority of points (67.1%) were plotted to within 51 metres of their true position.

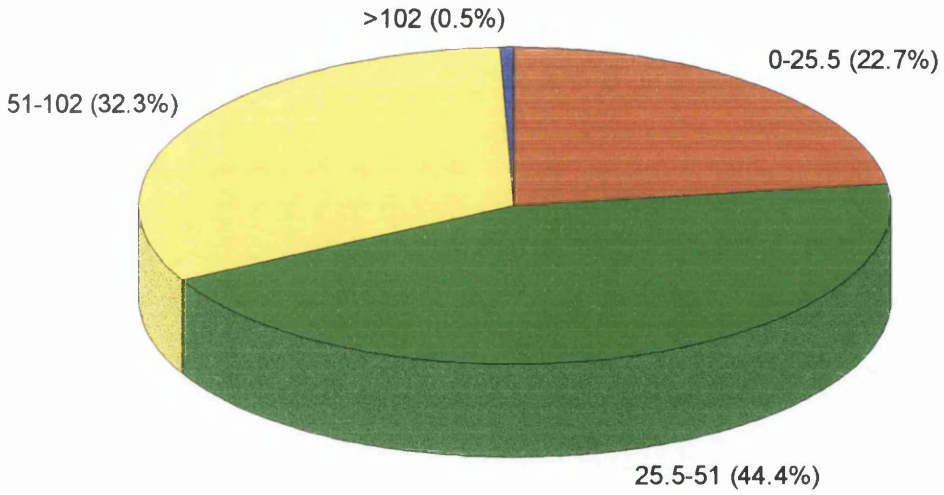


Figure 7.6 Percentage of Points Falling into each Buffer Zone

For the purposes of static testing the data was plotted on a map of 1:1250 scale, where map data is accurate to 0.4 of a metre (Ordnance Survey, 1996) and so positional errors of 102 metres were very apparent. However, the mountaineer would typically be using map data of 1:50000 scale, where the cartographic processes of displacement, exaggeration and generalisation can create errors of up to 50 metres (e.g. a road that is plotted on a 1:50000 scale map and represented by a 1 mm wide line is showing the road to be 50 metres wide. This is obviously not the case, but if the road width was scaled accordingly it would hardly be visible. Therefore the road width is exaggerated for clarity at the expense of accuracy). For map data of this scale, a positional accuracy of +/- 102 metres means that the user's position will be plotted to within 2 mm of their true position on the map. This kind of accuracy may be suitable for everyday hillwalking but for navigating in more hazardous terrain, the user may use a larger scale map data set (e.g. 1:25000) where the inaccuracies of the GPS Driver and receiver will be more noticeable.
The obvious solution is to make the position information accurate enough for plotting at all major scales that are likely to be used by the hillwalker (e.g. 1:25000 or smaller in most cases). The major inaccuracy that occurs in the position information is created by Selective Availability in the GPS system and so the solution to the problem would be to remove or bypass its effect.

As discussed by Gilbert (1996c) the removal of selective availability from the GPS system is currently being considered by the US Department of Defence. This would increase the accuracy of the receiver to +/- 30 metres (95% of the time).

Alternatively, the use of Differential GPS (DGPS) could be considered as a means of correcting the effects of Selective Availability. DGPS consists of three segments: the reference station, mobile station and data link (Figure 7.7). DGPS operates on the concept that errors produced for one location will be the same as those produced at another location within a limited area, as long as both the reference station and the mobile station observe the same satellites.

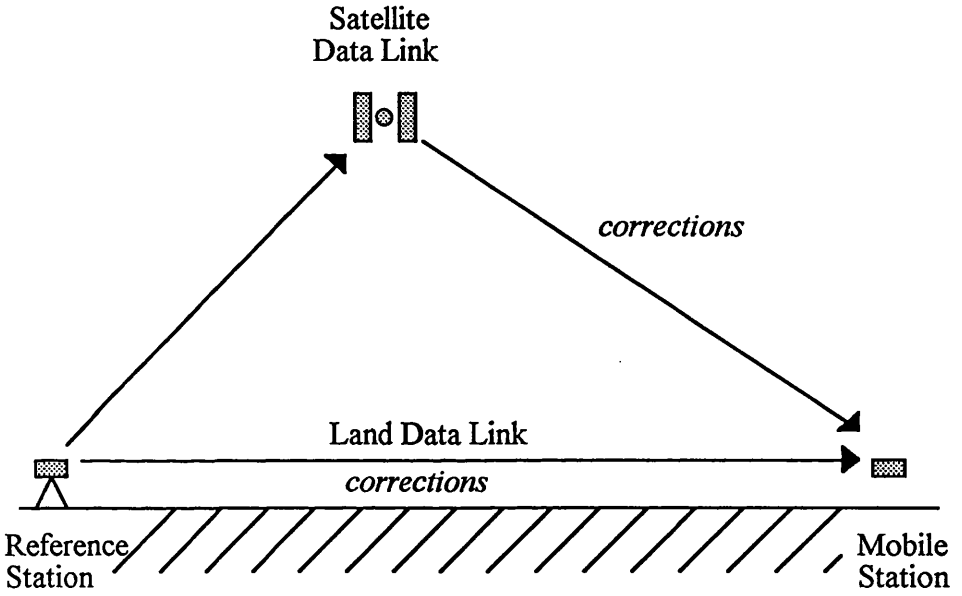


Figure 7.7 DGPS

The reference station is located at a known position and generates corrections for the positional errors induced by Selective Availability as well as other sources of error such as atmospheric conditions. The data link is then used to send or broadcast these corrections so they can be received by the mobile station. The data link can either be land based (e.g. radio transmissions or cellular phone networks) or satellite based (e.g. INMARSAT communications satellite (Seeber, 1993)). These corrections can increase the accuracy of the GPS receiver to 5 metres, depending on the quality of the corrections.

Applying this system to the Electronic Navigation Environment would at first seem an ideal way of increasing the accuracy of the position information it could provide. However, problems do exist with the system that mainly relate to the data link.

When operating in mountain environments the strength and quality of the transmitted data link is dependent on the type of link used. Low frequency radio links are capable of providing a wide area of coverage and possess good wave propagation and so the masking

effect that mountains have on radio signals would not be a great problem. However such systems are expensive to set up and the update rate (interval at which the corrections are transmitted) is low. High frequency radio links and cellular phone networks are capable of covering a smaller area and have poorer wave propagation characteristics and so are more susceptible to the masking effects of mountains. However they provide a faster age of correction and are less expensive to set up and run, and can even use existing radio networks (e.g. Classic FM).

Satellite based data links provide near global coverage and a fast age of correction. However such systems are expensive to establish and maintain even using existing satellite networks such as INMARSAT.

It is clear that Differential GPS would be a real possibility to solving the problem of positional accuracy in the navigation software, and would warrant further investigation that is outside the scope of this thesis.

7.3.5 Mobile Test Results and Analysis

The mobile tests were primarily used to test the operation of the GPS driver, as well as the operation of the Fix Interval function. As a secondary measure, the accuracy of the points was checked (although not as rigorously as in the static tests) to ensure no gross plotting errors occurred.

The GPS Driver proved to be successful when mobile in terms of operation as points were plotted as the car was driven round the test route (Figure 7.8). The repeated success of this test over a number of days meant that the GPS Driver could be deemed operational in real time (See also Appendix VI).

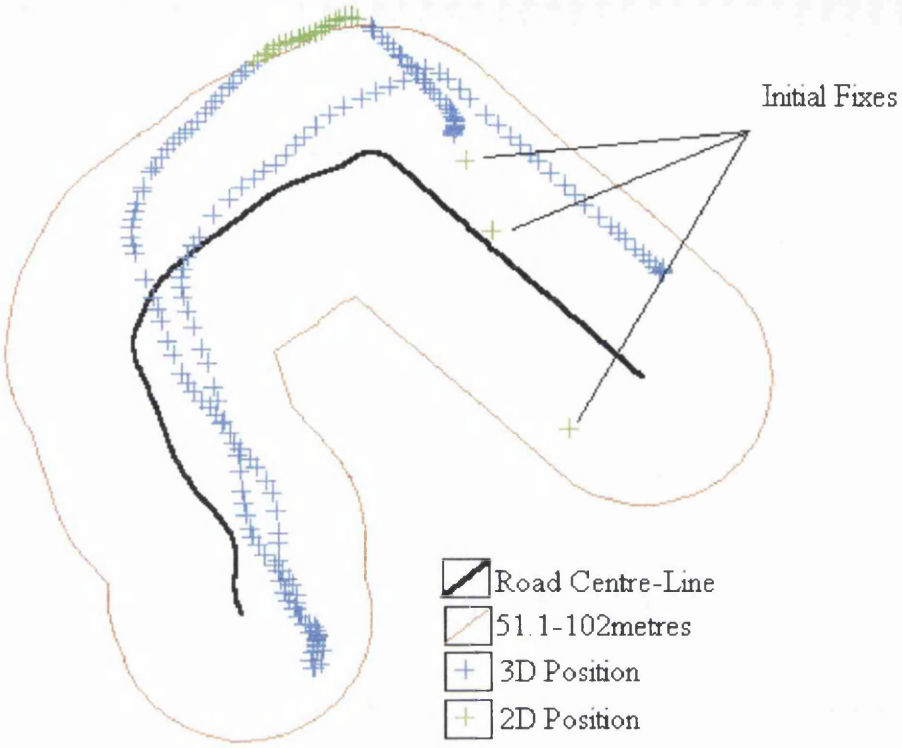


Figure 7.8 Mobile Test Results (Test 2)

The mobile accuracy of the Driver was checked by setting up a 102m buffer zone parallel to the road centre-line (Figure 7.8). The total number of points plotted for all mobile tests was analysed to see what percentage of points fell into the buffer zone.

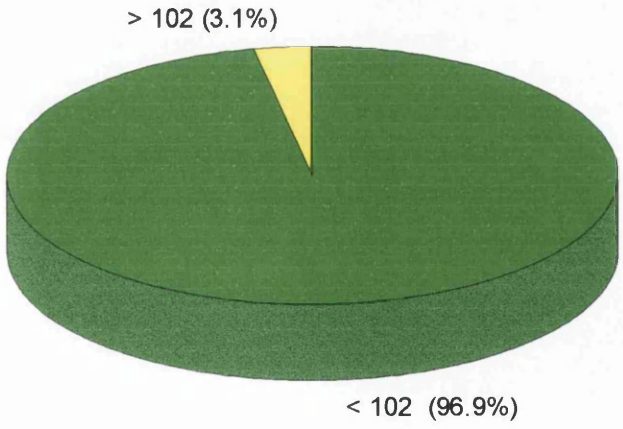


Figure 7.9 Percentage of Points Falling into the Zone

As can be seen in Figure 7.9 96.9% of all points tested were plotted to within specified accuracy. The higher percentage of points lying outside the tolerance than in the static tests

was due in part to the mode of operation of the receiver switching from three-dimensional to two-dimensional for the same reasons as those discussed for the static tests (i.e. internal set up of the GPS receiver). The initial 3 positions gained during Test 2 were obtained in two-dimensional mode which reduced their accuracy. This occurred again when the receiver was mobile and operating in three-dimensional mode, and then lost lock on one of the satellites, whereby the receiver reverted back to two-dimensional mode (Figure 7.8).
The potential for reduced accuracy positions gained whilst the receiver is in two-dimensional mode is clearly cause for concern as far as the mountaineer is concerned. Initial fixes during the tests were often determined using two-dimensional mode and the test results show that this can produce inaccurate position fixes. The problem could be overcome if the mask criteria of the GPS receiver were tightened, or the Electronic Navigation Environment checked the positions it receives and only plots those gained using the three-dimensional mode.

Once the basic operation and accuracy of the GPS Driver had been established, the additional functions of the GPS Driver could be tested. This included testing the fix interval, fixed map and moving map displays.

(i) Fix Interval
The fix interval was tested by setting it at 5 seconds and then driving round the test circuit. The data was recorded and plotted and the time between each fix measured using a stop watch. With points being plotted every five seconds, the fix interval function was deemed operational (Figure 7.9). This was repeated for a ten second fix interval and once again found to be successful (Figure 7.10).

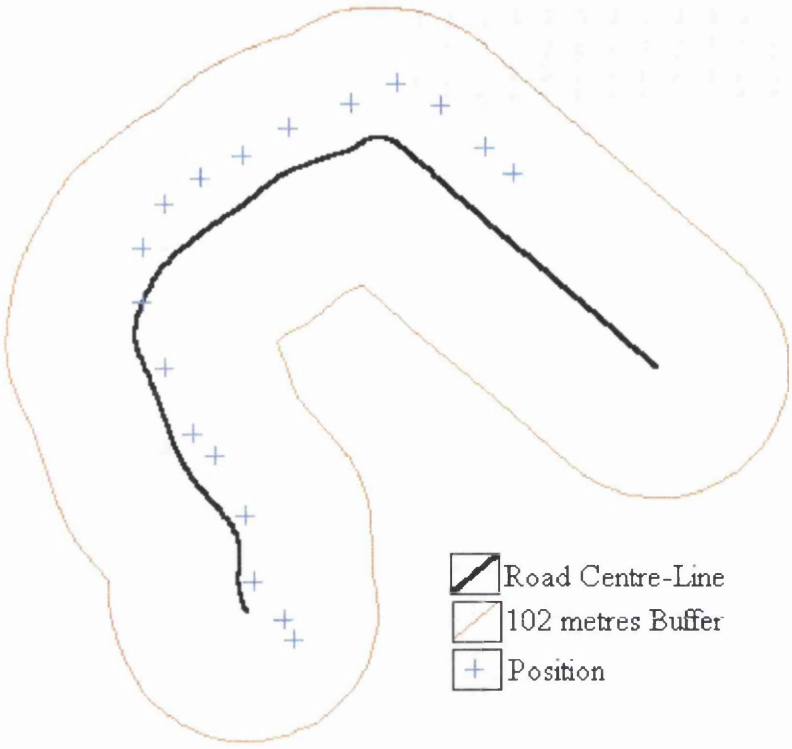


Figure 7.9 Five Second Fix Interval Test Results

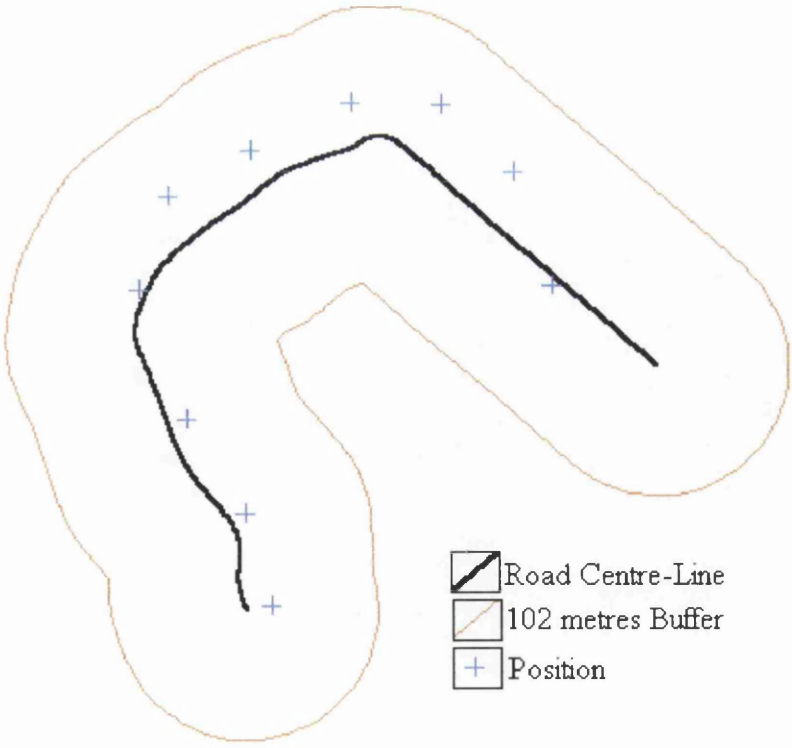


Figure 7.10 Ten Second Fix Interval Test Results

(ii) Fixed Map Testing

For Fixed Map Display Testing, the GPS Driver was activated and run, plotting points on both raster and vector map data to ensure it operated for both types in real time. This display type was used for all static and mobile tests and was deemed fully operational.

(iii) Moving Map Testing

The moving map display operated with both vector and raster data, but not as originally envisaged. This type of display was supposed to scroll the map around the users position which would be fixed in the centre of the display. However, it soon became apparent that Map Basic was not capable of this and instead redrew the whole display with the position in the centre. This meant that each time a new position was to be plotted, the screen 'flashed' as it redrew the display. When using vector data, this typically took less than 1 second, and so although the map 'flashed' it was not a great inconvenience. However, when using raster data, redrawing could take up to 7-8 seconds. This meant that at fix intervals set below 20 seconds, there was barely time to view the position before the display was redrawn.

Under normal walking conditions of clear weather and non-hazardous terrain, the typical walker may only require a fix interval of thirty minutes to an hour, or will use the system for single fixes. Even if the terrain were hazardous and the weather poor, the user would be unlikely to require a fix interval of less than two minutes. Therefore, this problem, whilst aesthetically annoying, would not present a huge disadvantage to the typical user.

This problem could be overcome if the map display could be scrolled or processing power was such that any redraw would take at most 1-2 seconds.

7.4 Tools Testing

Tools testing was carried out to ensure that the tools operated as part of the navigation software system. The tools were already tested at modular level and deemed operational independently.

System testing of the tools involved using each tool in turn as part of the Electronic Navigation Environment. If it carried out the operation intended it was deemed operational. Any problems would be highlighted through repeated use of the tools in the following situations:

- Operate Tools with vector data
- Operate Tools with raster data
- Operate Tools with GPS Driver running
- Operate Tools with no map data displayed

Table 7.8 shows those Tools tested; those that operated as desired are marked with a tick, and those that had difficulty in operating are marked with a cross and discussed below.

	Operation			
Function / Tool	**No Data**	**Vector Data**	**Raster Data**	**GPS Driver**
GPS On	✓	✓	✓	✓
GPS Off	✓	✓	✓	✓
Fix Interval	✓	✓	✓	✓
Advanced Options	✓	✓	✓	✓
Display Compass	✓	✓	✓	✓
Hide Compass	✓	✓	✓	✓
Zoom In	✓	✓	✓	✓
Zoom Out	✓	✓	✓	✓
Move Map	✗	✓	✓	✓
Recentre	✗	✗	✗	✗
Undo	✓	✗	✗	✗
Plot WayPoint	✓	✓	✓	✓
Plot Route	✓	✓	✓	✓
Measure	✓	✓	✓	✓
Delete	✓	✓	✓	✓
Select	✓	✓	✓	✓
Estimate Journey Time	✓	✓	✓	✓
Open File	✓	✓	✗	✓
Close File	✓	✓	✓	✓
Save File	✓	✓	✓	✓
Register Map	✓	✓	✓	✓
Adjust Map	✓	✓	✓	✓
Help	✓	✓	✓	✓
Exit	✓	✓	✓	✓

✓ Tool operated correctly

✗ Tool malfunctioned or did not operate as planned

Table 7.8 Tools Testing Results

As can be seen from Table 7.8, most of the tools and functions operated correctly; however there were problems with several functions.

7.4.1 Map Move Problems

The problems with the Map Move tool occurred when it was used with no map data. When the user selected the Map Move icon and no map data was present, then a message box should have appeared in the map display window stating "A map sheet must be opened before using this tool". When the test was carried out, the map display window flashed, and the user had to click on the window in order for the message to appear (Figure 7.11).

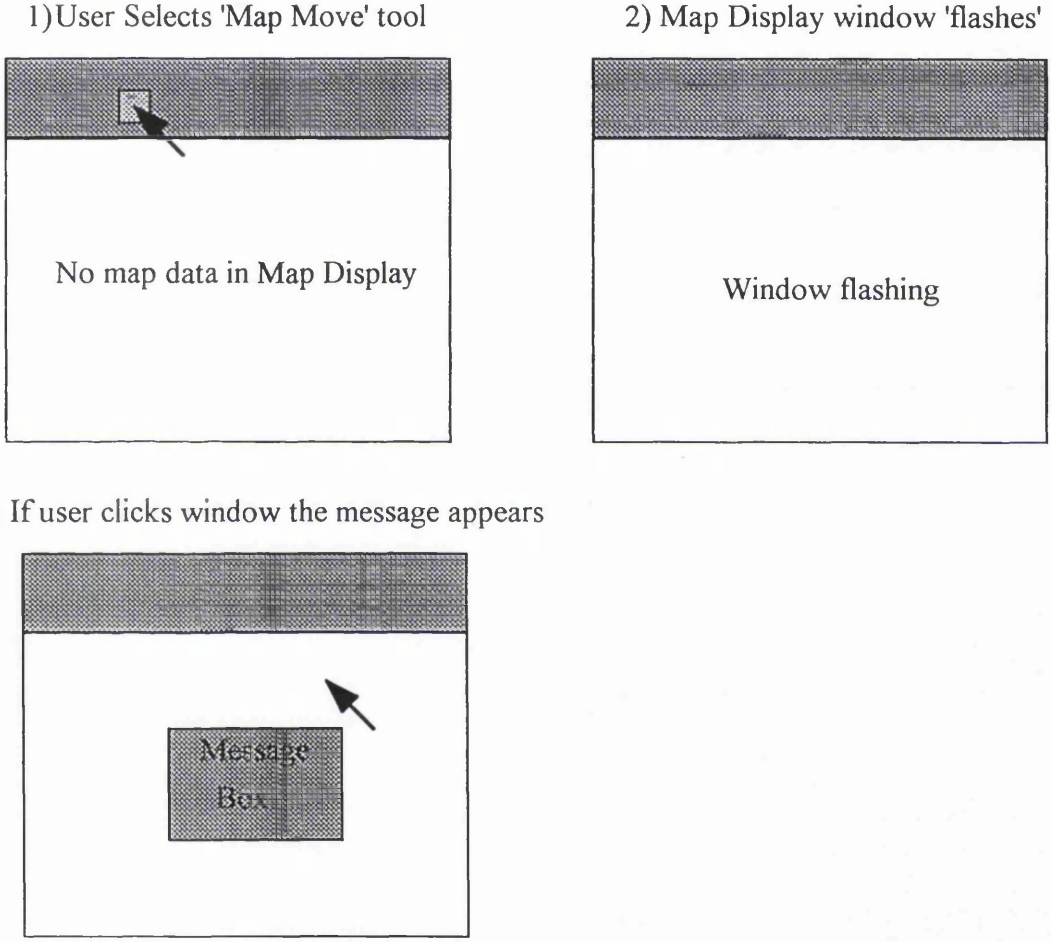


Figure 7.11 Map Move Error

Clearly the message box should have appeared automatically, instead of the user having to click on the Map Display to make it appear. The root of this problem lay in the fact that the Electronic Navigation Environment was developed in two different languages, and, in order to display the full system on the screen two different windows had to be created, one for the Map Display and one for the SUI. Clicking on the 'Map Move' tool with no data meant that the SUI window was active. In order for the message to appear in the Map Display, this window had to be active. If the Map Display was not active, then it flashed to let the user

know that an instruction had been sent to the window but, until it was activated, this instruction could not be turned into an action.

The solution to this problem was to use the 'Appactivate' statement in the SUI. This statement activates a window and was used to avoid the above problems occurring in the tools written using the 'plug-in' MapBasic procedures (e.g. Zoom In, Zoom Out, Delete, Select etc.), but was missed when coding the Map Move tool. The trapping and rectifying of this error reinforced the need for systems testing as modular testing overlooked this error.
When implemented into the Map Move tool this statement operated as required and the tool was deemed operational.

7.4.2 Recentre Problems

The problem with the Recentre tool was identical to that found in the Map Move tool. The same solution was implemented and the Recentre tool deemed operational.

7.4.3 Undo Problems

The Undo command was introduced after SUI testing and was intended to undo any action that the user carried out and wished to reverse. The Mapinfo Undo procedure was used but when tested would only undo edits carried out on the map, such as plotting points or routes. An undo that would reverse every function would require an extremely complex procedure, checking what action had been carried out and then reversing it. To carry this out, each action performed by the user would have to be stored in case the undo command was invoked. Designing and developing such a procedure was deemed to be outwith the scope of the investigation.

7.4.4 Open File Problems

The Open File function opened a map window and then added the Position map layer so that GPS and route information could be plotted on the map (Figure 7.12).

1) The user opens a new map file.

2) The position layer is added so points can be plotted on the map.

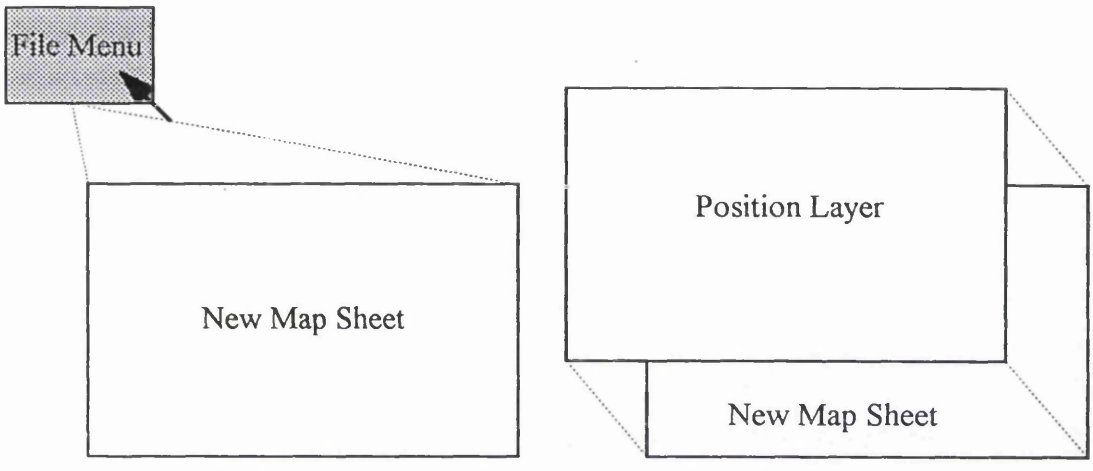


Figure 7.12 Open File Operation

This procedure operated correctly until an unregistered raster map file was opened. The procedure tried to add the position layer to the registration window that automatically opens when a new map sheet is loaded (the position layer is always added to the top-most window), causing the program to crash. The automatic appearance of the registration window was part of the MapInfo system which serves the same purpose as the Map Set Up function described in Section 5.3.3. This function could not be disabled, as its automatic appearance was intended as a safe-guard against the use of unregistered raster data.
A solution to the problem would be to allow the user to add the position layer manually, or for the program to 'detect' whether the layer should be added.

7.5 Software User Interface Testing

Diagnostic testing as defined by Sutcliffe (1991) was carried out on the Software User Interface (SUI). This meant that the main aims of these tests were to pinpoint poor design features. This involved testing two aspects of the SUI:

1) Ease of Use or Effectiveness: A measure of how well the interface performs; performance can be measured in terms of task completion within a target time.

2) Ease of Learning: A measure of how easy to learn the system is, which can be measured as: decreased error rates and decreased task completion time from the start of system usage. It should be noted that this is different to memorability, which is how well a user can remember the SUI after a time interval.

7.5.1 Software User Interface Test Procedure

The simplest and most effective way to determine whether these two elements were being satisfied was to measure the User Response Time (URT). The URT is the time taken by the user to complete a specific task or tasks. Mayhew (1992) suggests that between 6 and 20 subjects be used for testing depending on the size and complexity of the SUI and the nature of the tests. Rubin (1988) proposes that "For certain studies, those which have problem diagnosis as their central objective ... a small number of subjects can be entirely appropriate." Twelve subjects were used to test the SUI design, as it is relatively small and simple in terms of the options it displays (i.e. 24 options compared with over 100 displayed in, for example, Microsoft Works for Windows word processor).

In order to test Ease of Use, the URT was compared with a pre-determined target response time to ascertain whether the task could be completed efficiently. Each test subject was requested to perform a series of tasks and achieve particular goals using the SUI. The tasks chosen were carefully selected, to give a structured variety of menu depth and complexity.
To test the Ease of Learning, the tasks used to test Ease of Use were repeated in three cycles, with the Total URT for all tasks being measured for each cycle. When plotted graphically, the Total URT decreasing with each successive cycle would prove that the SUI was easy to learn. In summary the test involved the following elements:

(i) Establish tasks / goals to be completed.
(ii) Determine Target URT.
(iii) Record User Response Times for each task to determine Ease of Use.
(iv) Repeat above tasks twice to determine Ease of Learning.
(v) Questions and Queries.

This method was the most scientific method available, and using statistical analysis, small numbers of test subjects could be used without displaying bias (Rubin 1988). Other methods were investigated, such as questionnaires and rating scales (Rubin (1988), Sutcliffe (1995), Shneiderman (1987)). However, these methods rely heavily on the user judging how well the system works and are therefore prone to human error and poor judgement . It is also stated by Sutcliffe (1995) that these alternative methods are also time consuming to execute and can be difficult to analyse. Despite this it was recognised that suggestions and personal ideas would be useful for improvements or redesign.

(i) Establish tasks / goals to be completed

The first stage in setting up the test involved establishing a series of tasks that would suitably test all aspects of the SUI. As well as this, the test had to be short, to hold the subjects attention, as according to Sutcliffe (1995) "...too many tasks will test the tolerance of the [test subject]". The following five tasks were used in the test.

Task 1: "Zoom In on an area of the map."

This task examined the test subject's response to utilising Level One Tools, as well as interacting with the Map Display (Figure 7.13). Because they are Level One Tools, they should produce the fastest response time during testing.

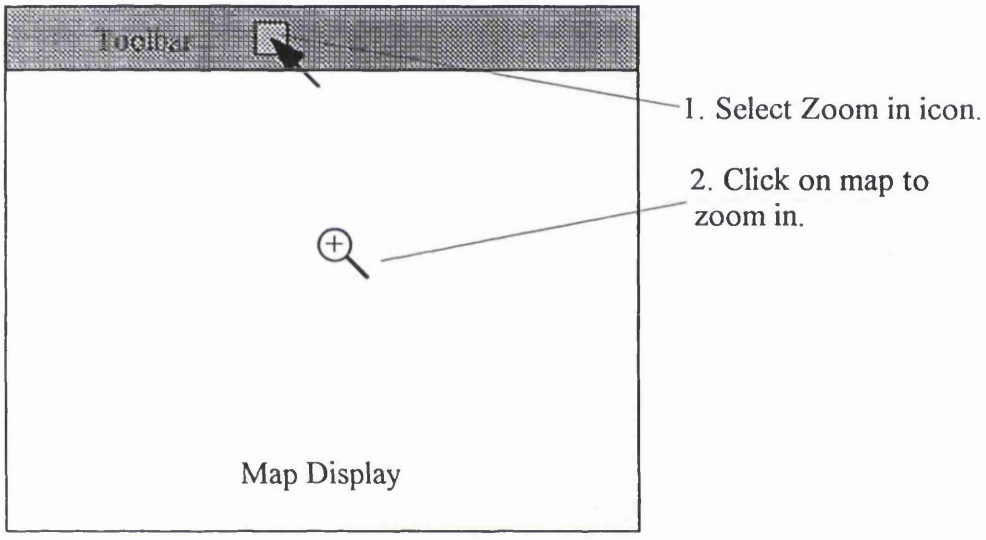


Figure 7.13 Task 1

Task 2 "Select the Route Planning icon and plot a route on the map."

This task required the use of a Level Two function, and interaction with the Map Display. It was more complicated than Task 1 and so a more detailed description of the task was given to the test subject. The response time was expected to vary depending on how elaborate a route the test subjects plotted. This was taken into account by measuring only the time taken to select the tool and plot the first point on the route (Figure 7.14).

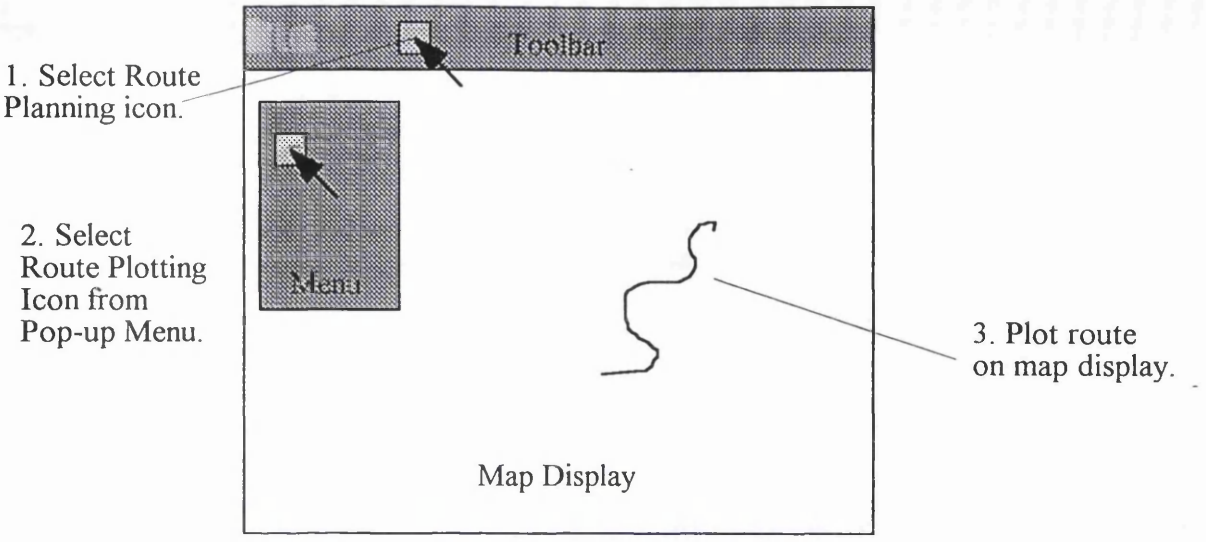


Figure 7.14 Task 2

Task 3 "Plot Waypoints at the beginning and end of your route."

This task was designed to build on the knowledge the test subject gained from Task 2. The Waypoint function is located in the same menu as the Route Plotting function, and the response time would gauge whether the test subject had noticed this or not (Figure 7.15).

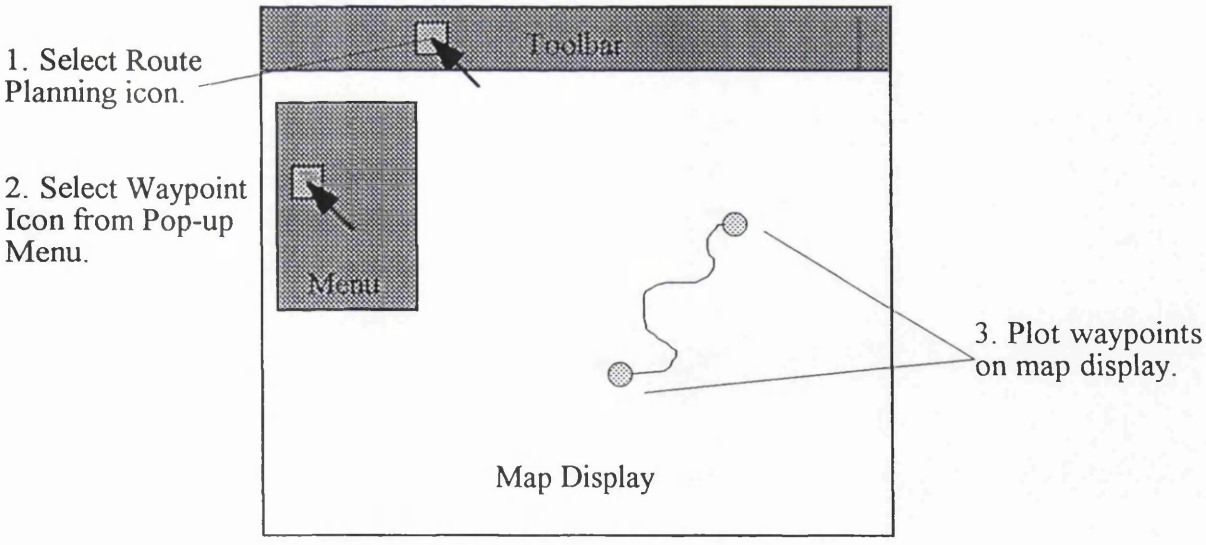


Figure 7.15 Task 3

Task 4 "Select the GPS Menu, turn on the GPS, select a five second fix interval and a fixed map display."

This task required the completion of several tasks in succession (Figure 7.16). As in Tasks 2 and 3, these functions were also located at Level Two of the menu, but used different

display methods from the previous two tasks at Level Two (i.e. Radio buttons and list boxes were also used, instead of just icons and command buttons).

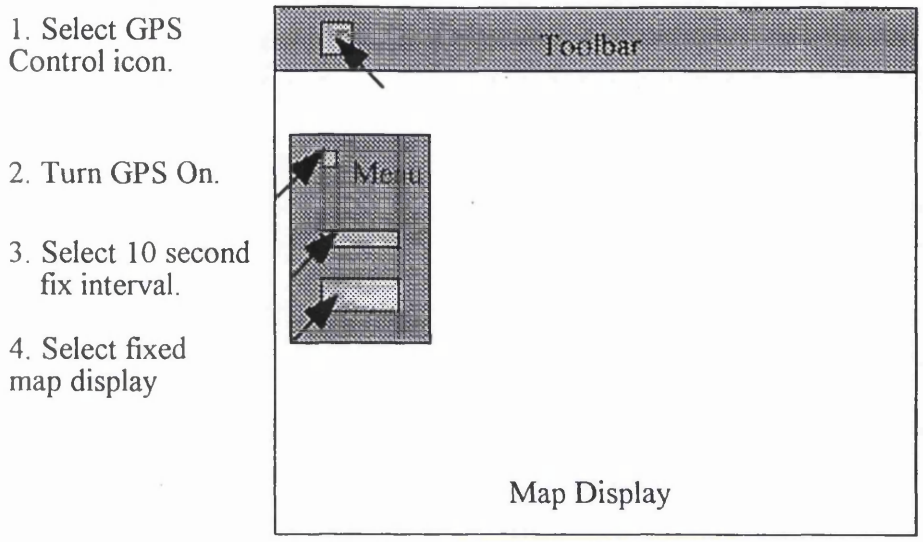


Figure 7.16 Task 4

Task 5 "Select the help page describing the zoom in tool."

This was the final task and was used to test a Level Three help option (Figure 7.17).

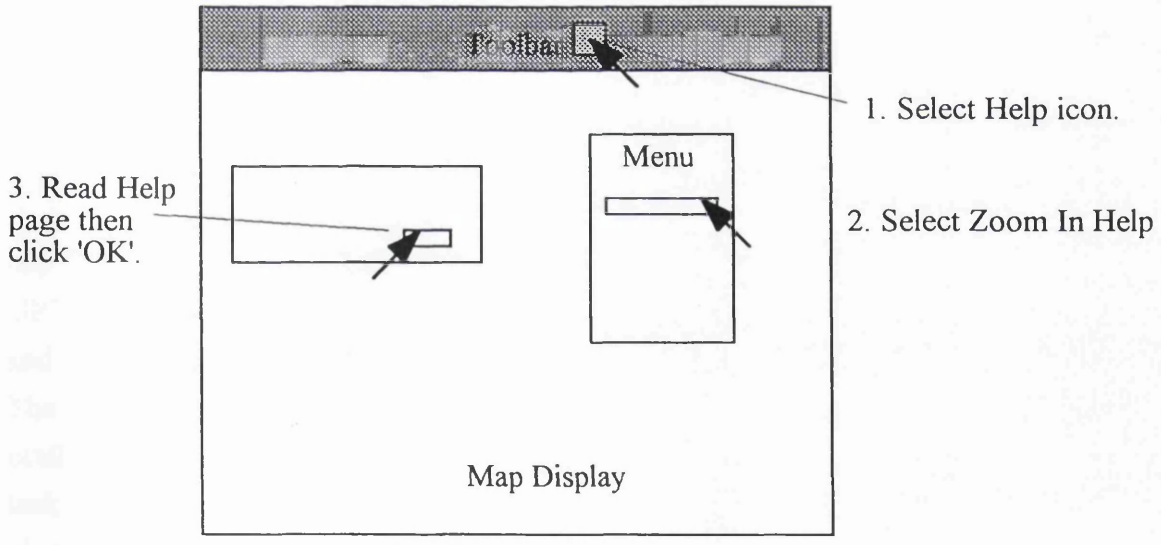


Figure 7.17 Task 5

These tasks were carefully chosen to test the scope of the SUI and reveal any deficiencies in its design.

(ii) Determine Target URT

This element was essential for the evaluation of the test results. A maximum duration had to be established for each task and was determined by taking into account the following factors, and using the guidelines set down by Mayhew (1992):

- The level in the menu system at which the task or tasks were located (1 second for each level).
- The selections that had to be made to complete the task (2 seconds for each selection to be made).
- Map Display interaction (2 seconds for any interaction).

The maximum response time should have given the Test Subjects adequate time for the functions to be activated or tasks completed (Table 7.9). Longer recorded times highlighted areas of difficulty or a lack of clarity.

Task	Menu Level	No. Of Selections	Map Display Interaction	Target URT (seconds)
1	1x(1)	1x(2)	1x(2)	5
2	2x(1)	2x(2)	1x(2)	8
3	2x(1)	2x(2)	1x(2)	8
4	2x(1)	4x(2)	-	10
5	3x(1)	3x(2)	-	9

(numbers in brackets indicate the seconds each criterion must be multiplied by)

Table 7.9 Target User Response Time

(iii) Record URTs for each task to determine Ease of Use.

URTs were measured on a stopwatch, to the nearest tenth of a second. The response times and any errors made were then recorded on a test sheet (See Appendix VII for specimen). The test subject was timed from the moment when the task was communicated to them orally. The test subject was requested not to raise any queries about how to complete the task; however, if the task was not understood, then a minimum of further explanation was given. This process was completed for each of the five tasks. Once all five tasks had been completed, the first test cycle was complete.

(iv) Repeat above tasks twice to determine Ease of Learning.

The above process was then repeated twice in order to complete cycles two and three.

(v) Questions and Queries

Once the formal test was completed the test subject was asked about any errors made and why they felt they made the error. This was to gain some insight as to why an error had been made - whether it was down to poor design or unfamiliarity with the SUI.

The test subject was also asked to browse the SUI inspecting and exploring the tools and functions on offer. Any comments or suggestions were noted, as well as those made during the formal test. This process was seen as an informal questionnaire, and hopefully the test subject would discover any poor design or suggest improvements that could be made.

7.5.2 Software User Interface Test Results and Analysis

This section illustrates and discusses the results gained from the Software User Interface (SUI) test.

(i) Ease of Use

The Ease of Use of the SUI was tested by comparing the Mean URT of each test subject for Cycles 2 and 3 of the test combined, to the Target URT value for each task. Cycle 1 was used to give the test subjects some familiarity with the SUI, as complete unfamiliarity would reveal lack of knowledge in the user, instead of poor functionality in the SUI.

If the Mean URT had exceeded the Target URT consistently then the tool tested would not have functioned as required and would have had to be redesigned.

For Task 1 (Figure 7.16), eleven of the twelve test subjects completed the task within the target time. Subject 3 took on average 22.9 seconds to complete the task over Cycles 2 and 3. The test subject had problems deciding how the tools worked during Cycle 1, which was used as a familiarisation exercise. It was clear this problem had not been resolved and continued during Cycle 2 (two interpretation errors were made trying to complete the task). However this was a user based problem as opposed to a SUI problem, as no other subjects recorded average URTs outwith the target time.

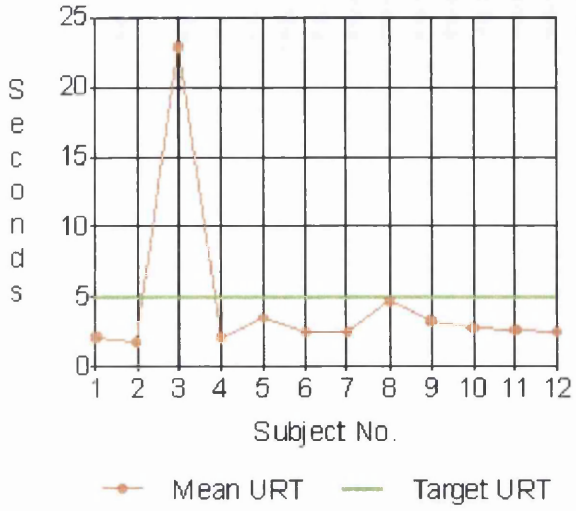


Figure 7.18 Task 1 User Response Time

Subject 9 completed Task 2 over Cycles 2 and 3 in an average time of 8.2 seconds (Figure 7.19). This meant that the task was completed 0.2 seconds outwith the Target URT. Such a small excess time was deemed insignificant, when taking into account the fact that no other tests for this task were completed outwith the Target URT.

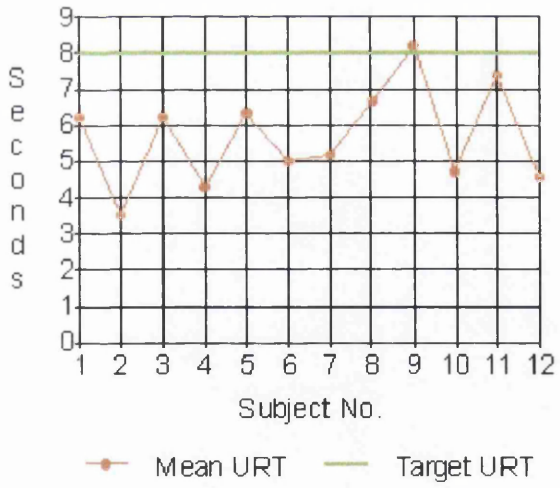


Figure 7.19 Task 2 User Response Time

For Task 3 (Figure 7.20), all test subjects completed the tasks within the target URT.

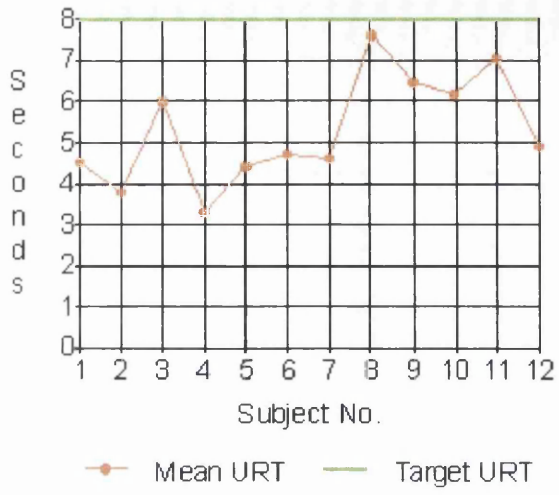


Figure 7.20 Task 3 User Response Time

The Task 4 (Figure 7.21) results show that subjects 3 and 5 completed the task outside the target time. This was mainly caused by a delay in the test subject selecting the appropriate icon (GPS control) or in the case of subject 5 an interpretation error being made in the selection of the icon. This highlighted an area of difficulty in interpreting this icon, and, as discussed during the questions and queries session, on the whole the test subjects felt that this icon was difficult to interpret.

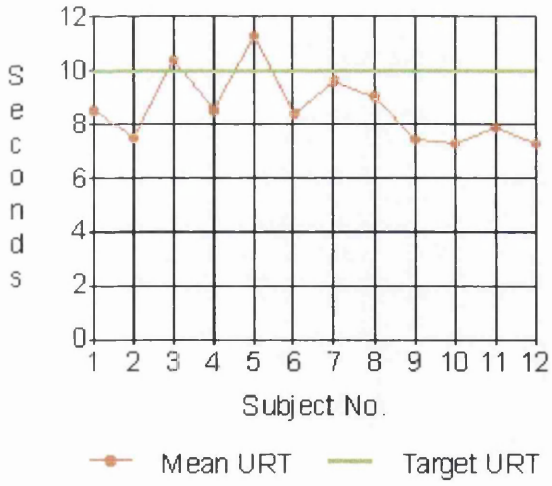


Figure 7.21 Task 4 User Response Time

For Task 5 (Figure 7.22), all test subjects completed the tasks within the target URT.

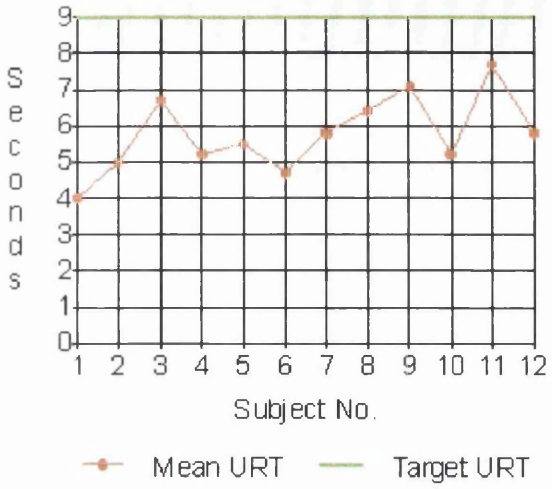


Figure 7.22 Task 5 User Response Time

These results showed that the tools tested were effective although, in the case of Task 4, a problem area was identified, which will be discussed later in this chapter. These tests revealed that no major redesigns were required and, since these tasks represented a cross section of the options the SUI displayed, the SUI was deemed to have operated effectively and be easy to use.

(ii) Ease of Learning

The Ease of Learning of the SUI was measured using two factors: the change in total time spent on all tasks in each successive cycle and the number of errors made in each successive cycle.

Figure 7.23 shows the average length of time all test subjects took to complete each cycle of the test as a percentage of the total average time for all the cycles combined. The average time spent on Cycle 1 was 49.5% of the total average time. This was the largest percentage of time spent on any cycle. This figure was expected due to the test subjects' unfamiliarity with the SUI at this stage.

During Cycle 2, 27.8% of the total average time was spent completing the tasks. This was a drop of 21.7% in the time taken between Cycle 1 and Cycle 2. During Cycle 3, 22.7% of the average total time was spent completing the tasks. This was a drop of 5.1% from the time for Cycle 2.

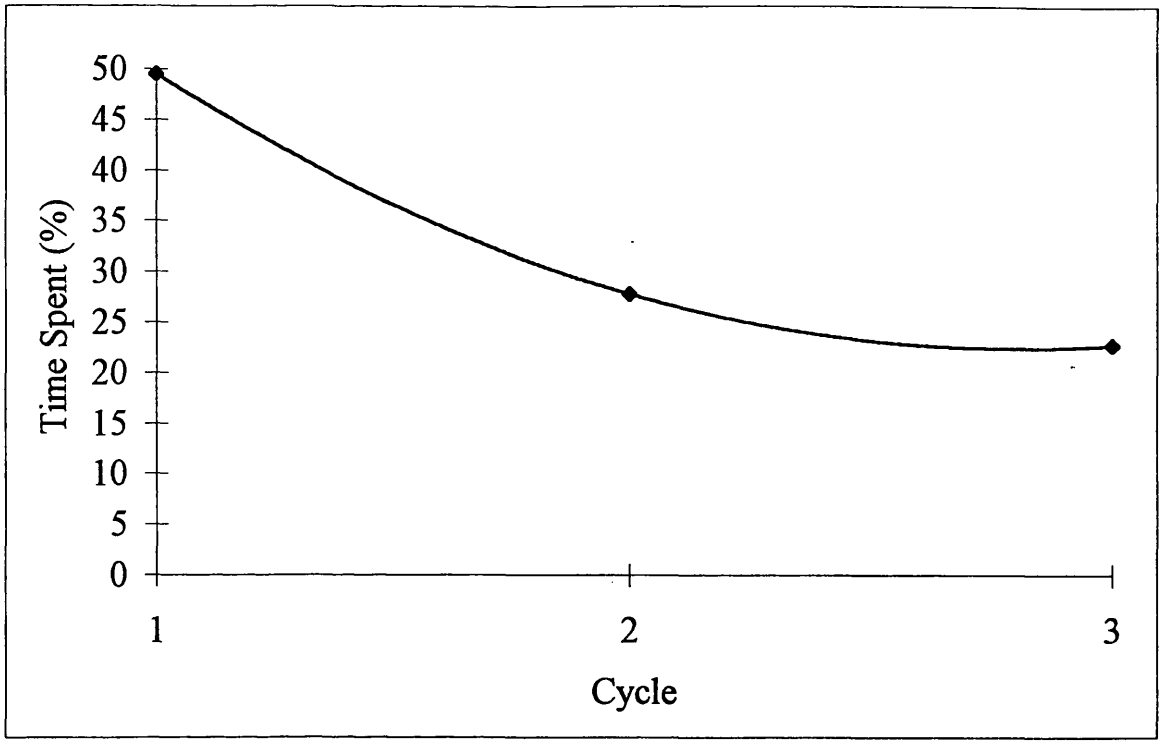


Figure 7.23 Average Time Spent on Each Cycle (%)

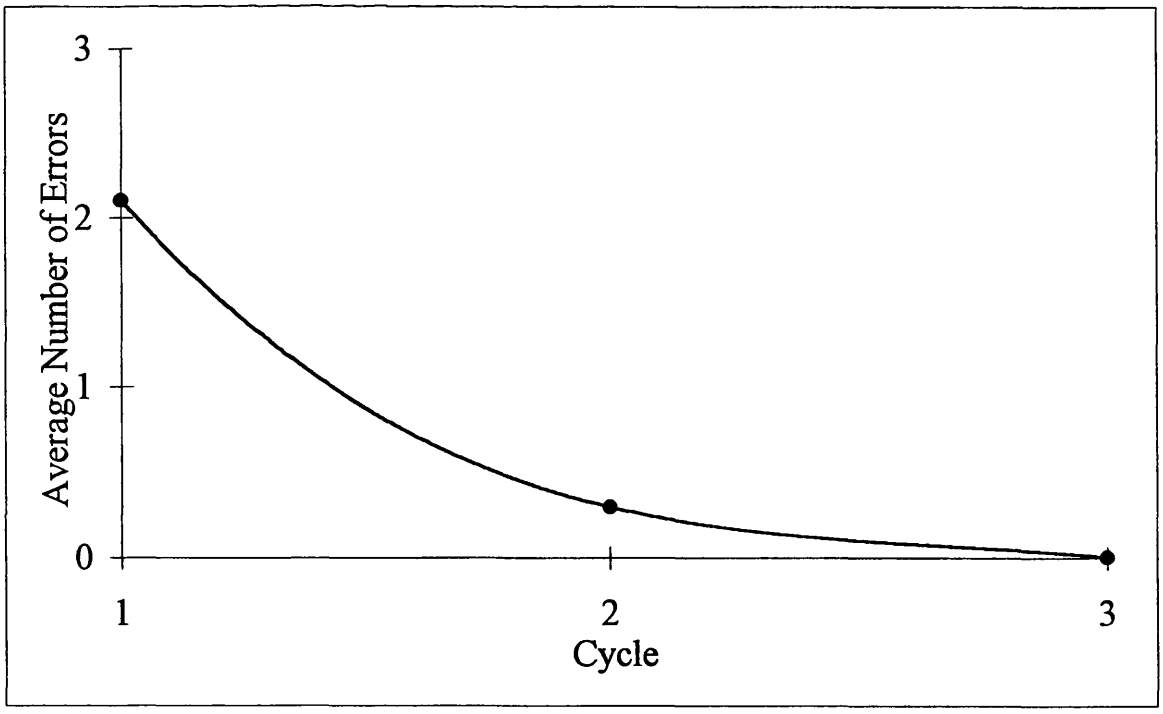


Figure 7.24 Interpretation Errors per Cycle

The large drop in time between Cycle 1 and Cycle 2 was expected as the initial part of the learning curve should show the largest improvement. The less pronounced drop in time from Cycle 2 to Cycle 3 signified the levelling out of the learning curve as, once the test subjects were fairly familiar with the tasks to be performed, further improvement on the completion of each cycle reduced. This pattern showed that the SUI was easy to learn and that after the first two cycles, the user quickly became familiar with the SUI.

This conclusion was reinforced by inspecting the number of interpretation errors made with each successive cycle. If the SUI were easy to learn, then the number of errors made should have reduced with each successive cycle.

Figure 7.24 shows the average number of errors made in each cycle. The graph clearly shows that the majority of errors were made during Cycle 1 when the user was least familiar with the SUI (an average rate of 2.1 errors made by each test subject). During Cycle 2, the average errors per person dropped to 0.3, showing a marked improvement in the interpretation of the SUI as the user became more familiar with it. Cycle 3 shows an error rate per person of 0, as by this time all test subjects were familiar enough with the particular SUI operations not to make any selection errors.

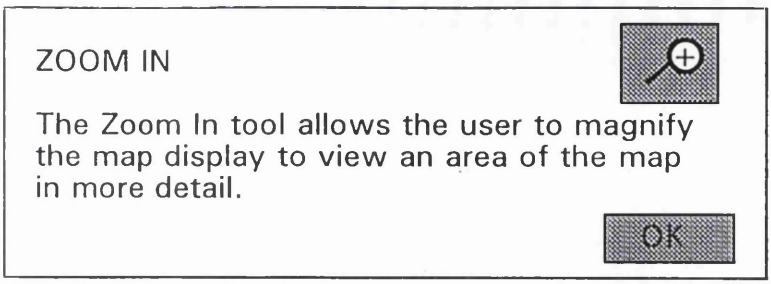
This analysis of the error rate per person per cycle reinforced the success of the SUI design in terms of Ease of Learning.

(iii) Questions and Queries

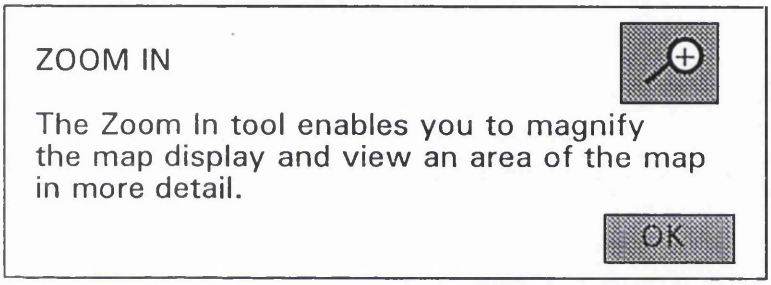
The test subjects were then asked to give suggestions as to how the SUI might be improved. The following suggestions were made:

- Include an 'Undo' command so that incorrect actions could be reversed. This arose from the fact that if an interpretation error was made and an incorrect action carried out, there was no way to reverse the action. This suggestion was made by 50% of the test subjects and so the relevant modifications were made to the navigation environment, as far as was possible (see 7.4.2).

- Personalise Help Menus. The original text in the help menus referred to the user in the third person. It was suggested that to refer to the user in the first person would make the help menus more user friendly (Figure 7.25).



Original Design



Redesign with personalised text

Figure 7.25 Example of Personalisation of Help Menus

- Add Icons to Help Menu. The original help menu provided a list of functions that the navigation environment could perform for which help was available. It was suggested that the relevant toolbar icon be placed next to each function name in the help menu, in order to assist the user in relating the help menu items to the toolbar functions (Figure 7.26). This improvement was also undertaken, as it was envisaged that it would make the system more user friendly.

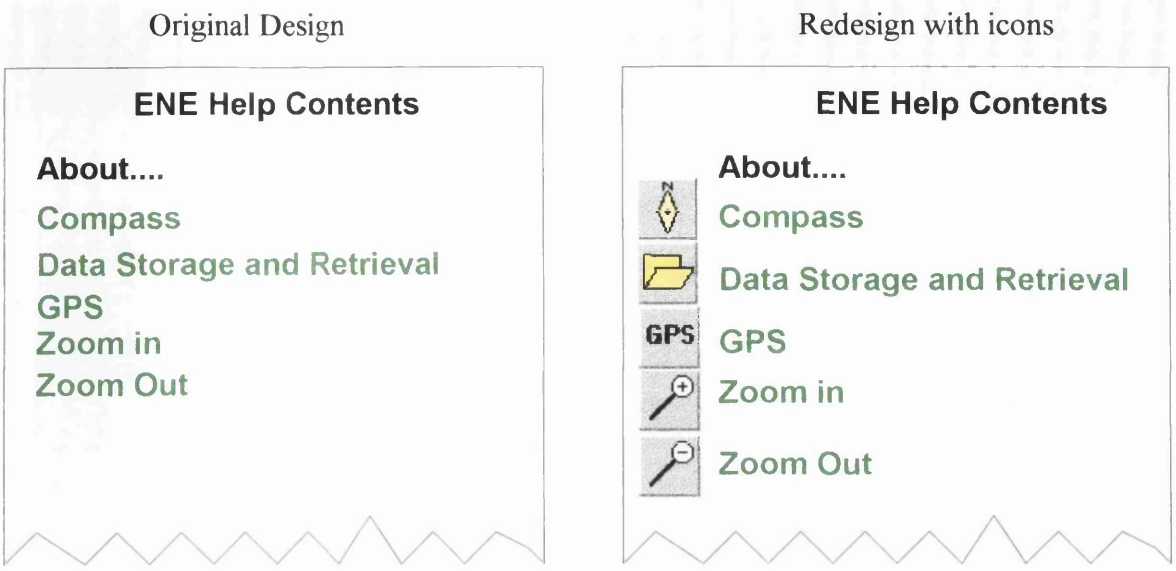
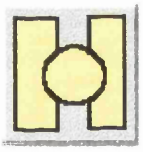


Figure 7.26 Adding Icons to the Help Contents

Suggestions were also made to change some of the icons used. The GPS icon came under scrutiny as 42% of test subjects searched for the initials 'GPS' when carrying out Task 4, and did not associate the satellite symbol with GPS. This icon was therefore changed to the text-based design (Figure 7.27).

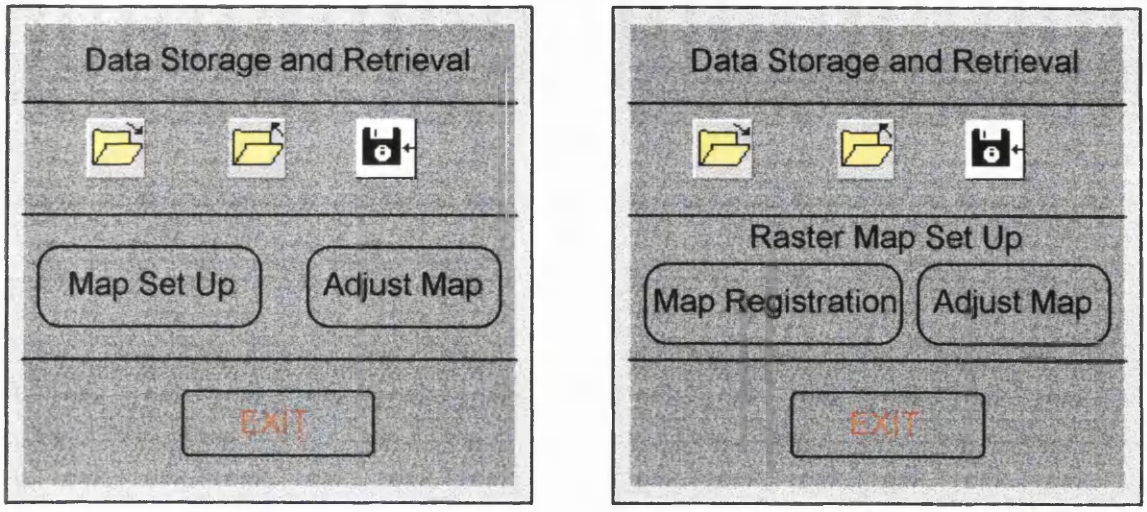


Original icon used a satellite to symbolise GPS.

This was redesigned using the 'GPS' initials.

Figure 7.27 GPS Icon Redesign

The final suggestion made was that the 'Map Set Up' command name was too easily associated with opening a new map file. This name was therefore changed to 'Map Registration', with a heading being added to the menu of 'Raster Map Set Up' to clarify the situation (Figure 7.28).



Original Design

Redesign to clarify command meanings

Figure 7.28 Map Set Up Redesign

7.6 Testing Summary

The tests carried out on the Electronic Navigation Environment were intended to assess whether or not the system met the user requirements and to diagnose any problems or errors that existed within the system.

The GPS Driver test results showed that the driver operated within its specified accuracies. The suitability of these accuracies with respect to the user will vary, depending on the map data they employ. Because of this, a method of achieving greater accuracy with the GPS receiver should be employed, such as Differential GPS, to allow the user to navigate with larger scale, more accurate data.
The operation of the GPS Driver was successful, and it interacted well as part of the navigation environment. The only real deficiency was the use of a moving map display with raster data, which is as much a problem with the processing power of the hardware as a limitation of the software. This situation may not even present a problem to the user, due to the relatively long fix interval the mountaineer would typically use. Even if short fix intervals were employed, the rapid increase in the processing power of hardware could compensate for this.

The Map Display testing showed that it operated as envisaged, but a problem was found when using colour raster map data. This problem highlighted the importance of making sure that the modules interacted successfully, as the root of the problem lay not with the map display itself, but its integration with the Electronic Navigation Environment. Although this problem was quite serious, possible solutions exist in either ensuring correspondence of the colour palettes of the Software User Interface and Map Display or fully integrating the modules of the navigation environment.

The tools testing showed that these functions operated successfully on the whole, but once again problems were revealed that were created by interaction with other modules. These problems however were not serious and were solved.

The SUI tests showed that the interface was easy to use and easy to learn which was an essential requirement of the user. Some problem areas were identified and alterations were made once the tests had been carried out, but these were relatively minor and there were no major faults with the SUI, or how it interacted with the other navigation environment modules.

On the whole, the tests revealed that the Electronic Navigation Environment would meet the requirements of the user as a navigational tool. However, a few minor improvements would greatly increase the software's suitability to the task.

CHAPTER 8: CONCLUSIONS AND FUTURE DEVELOPMENTS

8.1 Discussion

From the research carried out for this thesis, many conclusions can be drawn on the design, development and testing of the Electronic Navigation Environment.

1) The following points were established regarding the background of the Electronic Navigation Environment:

- Current navigation techniques rely mainly on the map and compass for position determination and orientation, and the ability of the navigator to relate their position on the Earth to that on the map.

- Whilst many mountaineers and hillwalkers use these tools successfully, the figures produced by Anderson (1995) suggest that many mountaineers are deficient in using the tools or are reluctant to use them.

- These tools and their related techniques cannot operate in all weathers, 24 hours a day. For instance, position determination at night is useless unless sighted points are illuminated. Similarly, trying to read a map in driving rain or snow is very difficult. This can cause great inconvenience or even danger to the mountaineer.

- The recent use of GPS and the electronic compass by the mountaineer provides an excellent opportunity to integrate these elements with a computer and maps in a digital format, to create a personal navigation device. Such a device would provide the mountaineer with an easy to use, 24 hour all-weather navigation system in one single unit. The main aim of the thesis was to investigate, design and develop personal navigation software that would meet the requirements of the mountaineer or hillwalker.

2) Once these points had been established, the software was designed and it was revealed that:

- Conceptual design was a vital stage in the overall design process as it focused the requirements of the software specifically on the needs and actions of the mountaineer/ navigator. This meant that from the onset of the design process, the needs of the user

were being addressed directly which, according to Shneidermen (1987), is how any piece of software should be designed.

- Four modules were required by the Electronic Navigation Environment: the Map Display, GPS Driver, Tools and Software User Interface.

- The selection of the equipment and software with which to design and develop the Electronic Navigation Environment had a very significant influence. Essentially, the design of the navigation environment proved to be a compromise between:

  User Requirements - what the user wants from the software
  Software Requirements - what the software requires to make it functional
  Development Equipment/ Software - the flexibility and/or limitations of such equipment and/ or software

  One example of this was the map registration process. This was required whenever a new raster map was opened and was a requirement of the MapInfo system. Because MapInfo was used as the basis for the Map Display, this process had to be implemented into the Electronic Navigation Environment. It could therefore be seen as a software requirement, brought about by the functions presented by the development software. However, the process of map registration was seen as a complication of the Electronic Navigation Environment for the typical user.

3) Once the navigation software system had been designed, it was developed. From the development of the software the following conclusions were drawn:

- The design of the Electronic Navigation Environment was, on the whole, successful, as few redesigns were required. All of the redesigns that took place were fairly minor. It is highly likely that any design and development process will require some aspect of redesign during the development stage.

4) The main aim of the investigation was to develop a fully functional piece of software capable of operation in a personal navigation device. This was essentially achieved as the navigation system provided the following functions that were fully tested for errors and operational problems:

- A digital map display capable of displaying a wide range of map data in a display made as large as possible.

- A GPS driver to provide 24 hour all weather positioning, which plotted the users position graphically on the map display in real-time, as well as storing positions in a database.

- A range of tools which equated to conventional techniques. The tools that the Electronic Navigation Environment provided allowed the user to carry out route planning, time estimation, map manipulation and GPS control, as well as providing facilities for the user to store and retrieve data.

- An effective, easy to learn Software User Interface through which the user could interact with the navigation environment.

5) Systems testing illustrated that the Electronic Navigation Environment was, on the whole, operational and met the requirements of the user. However, it also revealed some limitations in the navigation environment:

- The accuracy of the GPS Driver was largely dictated by GPS system's accuracy. The specified accuracy of +/- 102 metres could only meet the users requirements when small scale (e.g. 1: 50 000 or smaller) map data was used.

The problems associated with the display of raster data and its affect on colour meant that whilst raster data could be displayed, it appeared corrupted. It was envisaged that the main type of data the mountaineer or hillwalker would use would be raster, and so this was a fairly major limiting factor.

6) Finally it should be considered that the main reason for carrying out the tests was to find any errors or problems with the software. Whilst the tests went some way to assessing the fulfilment of the user requirements, a more comprehensive assessment of the Electronic Navigation Environment was not possible without the development of the hardware aspect of the personal navigation device, followed by fully integrated field tests.

Possible solutions were found to these limitations and are proposed as future developments.

8.2 Future Developments

(i) Mapinfo 3.0 to Mapinfo 4.0 Upgrade

Upgrading the Mapinfo GIS system used in the Electronic Navigation Environment from version 3.0 to 4.0 would provide many advantages and enhancements for the system. Most importantly, version 4.0 has the ability to fully integrate itself with Visual Basic programs. This would remove the need for Dynamic Data Exchange links and make communication between the various modules extremely simple. It would also solve many of the problems that were revealed during the development and testing of the software such as the colour problems encountered when using raster data. The increased functionality provided by MapInfo 4.0 would also give greater freedom and scope to design better position and waypoint symbols. Also, version 4.0 allows the rotation of symbols and so the position cursor would also be able to show direction of travel.

(ii) Investigation into DGPS for the Electronic Navigation Environment

One of the main limitations of the Electronic Navigation Environment was the accuracy of the GPS data. It was discussed during GPS testing that increased accuracy of the GPS position through the use of Differential GPS could be a possible solution. Increased accuracy of the positional coordinates would allow the navigation environment to be used on a wider range of map data at varying scales.

(iii) Integration of an Electronic Compass into the Electronic Navigation Environment

The integration of an electronic compass would be another possible future development. The orientation of the user is an extremely important aspect of the navigation process, which could not be fully explored due to the unavailability of such a device during the development of this project.

(iv) Design, Development and Assessment of a Personal Navigation Device

As mentioned in chapter 2, the software investigated in this thesis was designed to be part of a personal navigation device. Therefore the final proposed future development would be to design and build the hardware for the device, and integrate the hardware with the software developed in this thesis. Only once the software is being run on such a device can its true effectiveness in relation to the user be assessed.

8.3 Conclusion

The main conclusion that can be drawn from this investigation is that it is feasible to design and develop personal navigation software capable of emulating the tools traditionally used by the mountaineer. Even though the Electronic Navigation Environment possessed some limitations, the research has clearly illustrated that these limitations can be overcome.

It can also be concluded that the electronic medium provides the opportunity not just to overcome the inherent deficiencies and difficulties encountered whilst utilising the traditional tools of navigation, but to enhance the tools as well, and to integrate them into a single system - the Electronic Navigation Environment.

Appendix I Software Suitability

This table corresponds to Figure 4.1 shown in Section 4.2.1 and illustrates how the figure was derived.

Package	Type	MapData	Development	Coordinates	Layered	Route Info	Comms.	Total	Suitability (%)
Mapinfo	GIS	3	3	3	3	3	3	18	100
Atlas	GIS	1	3	3	3	3	3	16	89
Surfer	GP	3	1	3	3	0	0	10	56
Mapviewer	GP	3	0	3	1	3	0	10	56
DeskMapper	CAD	1	1	3	3	3	0	11	61
MapVision	GP	1	0	3	3	3	0	10	56
LView Pro	GP	3	0	1	0	1	0	5	28
Lanscape Expl.	GP	3	0	3	1	0	0	7	39

* *GP = Graphics Package*

Each package was rated on whether it met the requirements as a basis for the ENE.
A rating of 3 was given if a package could meet a particular requirement fully. A rating of 1 was given if a package could meet particular requirement, but in a limited fashion. A rating of 0 was given if a package had no functionality
to meet a particular requirement.

Appendix II DOS ASCII File Links

Appendix II DOS ASCII File Links

DOS ASCII (American Standard Code of Information Interchange) File links make use of a program's ability to write information to a file in order to exchange information (British Computer Society, (1995). With respect to the electronic navigation environment, the module sending the information would write data or commands to an ASCII file and the module receiving the data would read from the ASCII file (See Fig I).

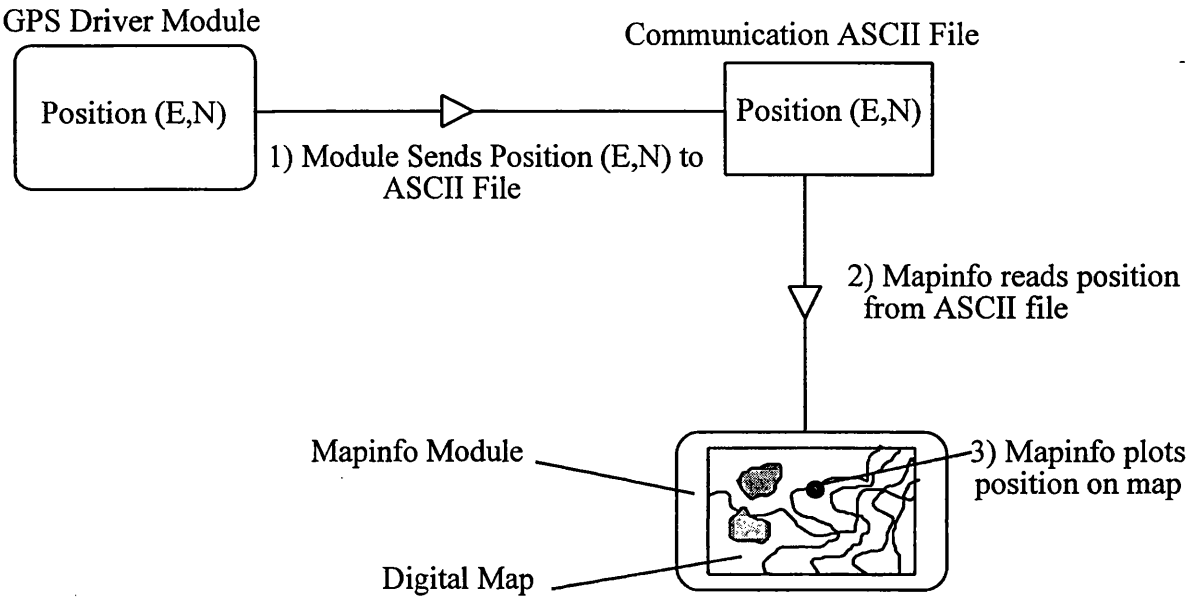


Fig I ASCII File Links

This method of linking the modules would be very simple conceptually and easy to implement, as all the modules that constituted the navigation environment were capable of writing and reading to ASCII files. Despite the method's ease of use it had two main drawbacks. The Electronic Navigation Environment was designed to run in a Windows environment, but the use of ASCII File links would mean that the DOS operating system would have to be used in order to access the communication files. The use of DOS would have slowed down the speed of operation of the program to such an extent as to be detectable by the user. The second drawback to this method would have been creating the link between GPS Driver module and GIS Module. This link had to be able to operate in real time, and evidence showed that opening and closing the communication file could not always be ensured to coincide with the flow of data. Non-coincidence of these two processes would have lead to the program crashing and so an unstable platform would have been created. To summarise the main problems with this method were:

- Unstable when handling real-time data
- Slow to process communications links

Appendix III Transformation Methods

(i) Molodensky Method

Source for stages 1-3: Breach M (1993)
Source for stage 4: Swann CP (1989) http://dcup1.cs.york.ac.uk:6666/fisher/software/coco/fortran Internet Website

The first method to be considered for the transformation takes place in four stages. Stage one requires a conversion from WGS84 Geodetic coordinates to WGS84 cartesian coordinates by means of the following equations:

$$^{WGS84}X = \nu Cos\phi Sin\lambda$$
$$^{WGS84}Y = \nu Cos\phi Sin\lambda$$
$$^{WGS84}Z = \nu Sin\phi(1-e^2)$$

where:

$^{WGS84}X, ^{WGS84}Y, ^{WGS84}Z$ = WGS84 Cartesian Coordinates
ϕ,λ = Geodetic Coordinates

ν = radius of curvature at latitude ϕ perpendicular to a meridian:

$$\nu = a / Sqr(1 - (e^2 * Sin^2\phi))$$

e^2 = eccentricity of the ellipsoid:

$$e^2 = (a^2 - b^2) / a^2$$

Stage two involves a transformation from the WGS84 ellipsoid to Airy's Spheroid. This involves applying a series of shifts to the coordinates calculated in stage one.

$$^{OSGB36}X = ^{WGS84}X + 375$$
$$^{OSGB36}Y = ^{WGS84}Y - 111$$
$$^{OSGB36}Z = ^{WGS84}Z + 431$$

The coordinates are now in the reference ellipsoid that best fits the UK. Stage Three requires these coordinates to be converted to geodetic coordinates using the reverse procedure to that employed during Stage One.

$$^{OSGB36}\phi = (^{OSGB36}Z + e^2\nu Sin\phi) / (^{OSGB36}X^2 + ^{OSGB36}Y^2)^{1/2}$$

(N.B. The calculation of Latitude is an iterative procedure requiring a provisional value of Latitude. Convergence is usually achieved after 2-3 iterations.)

$$^{OSGB36}\lambda = \arctan\left(^{OSGB36}Y / ^{OSGB36}X\right)$$

Finally, these geodetic coordinates must be converted to OSNG. This involves the introduction of the Transverse Mercator Projection into the transformation process:

$$^{OSNG}N = m0 * (F - (P^2) / 2 * (-\nu * Cos^{OSGB36})\phi * Sin^{OSGB36})\phi - 100000$$

$$^{OSNG}E = m0 * ((P * \nu * Cos^{OSGB36})\phi - P * (P^2) / 6 * (-\nu) * Cos^{2OSGB36})\phi * (Cos^{2(OSGB36)}\phi) * (\nu / p - Tan^{2OSGB36})\phi) + 400000$$

where:

$a = 6377563.396$

$b = 6356256.91$

$m0 = 0.9996012717$ (Scale factor along central meridian

$\nu = a / Sqr(1 - e^2 * Sin^{2(OSGB36)}\phi)$

$p = \nu / (1 - e^2 * Sin^{2(OSGB36)}\phi * 1 - e^2)$

F = Constant polynomial value

To summarise, this method operates as follows:

- Convert from WGS84 Geodetic coordinates to WGS84 Cartesian coordinates
- Transform WGS84 to Airy by introducing a series of XYZ shifts
- Convert from Airy Cartesian coordinates to Airy Geodetic coordinates
- Convert Airy Geodetic coordinates to OSNG

(ii) Ordnance Survey Method

(Source: Ordnance Survey (1995a) and *Ordnance Survey (1995b)*

The second method was developed by the Ordnance Survey specifically for use in converting GPS satellite coordinates to OSNG.

The GPS coordinates are converted from WGS84 to OSGRS80 (Ordnance Survey Global Reference System 1980) which results in plane coordinates on the Transverse Mercator Projection. This means that Airy's spheroid is not used as is normal, but instead OSGRS80 is used as the reference ellipsoid for the UK.

The first stage in the calculation is to scale a and b by F:

$a = a * F$

$b = b * F$

then calculate Meridian Arc:

$$M = b[\{(1+n+5/4n^2+5/4n^3) * (\phi - \phi_0)\} - \{(3n + 3n^2 + 21/8n^3) * \sin(\phi - \phi_0) * \cos(\phi + \phi_0)\} + \{(15/8n^2 + 15/8n^3) * \sin 2(\phi - \phi_0) * \cos 2(\phi + \phi_0)\} - \{(35/24n^3 * \sin 3(\phi - \phi_0) * \cos 3(\phi + \phi_0)\}]$$

calculate Eastings and Northings values:

$$(OSGRS80)N = (I) + P^2(II) + P^4(III) + P^6(IIIA)$$

$$(OSGRS80)E = E_0 + P(IV) + P^3(V) + P^5(VI)$$

where:

M = Meridian Arc

n = (a-b)/(a+b)

a = semi major axis (6378137)

b = semi minor axis (6356752.3141)

F = Scale Factor on Central Meridian (0.9996012717)

ϕ_0 = 49 degrees:- True Latitude origin

η^2 = deviation of the vertical in the prime meridian plane:

$$\eta^2 = (v/\rho) - 1$$

ρ = radius of curvature of a meridian at ϕ:

$$\rho = v(1-e^2)/(1-e^2\sin^2\phi)$$

$P = \lambda - \lambda_0$ (λ_0 = 2 degrees:- True Longitude origin)
E_0 = False Origin (400 000)
N_0 = False Origin (-100 000)
I,II,III,IIIA,IV,V,VI = Polynomial Values

The next stage of the transformation involves applying a series of shifts to the coordinates calculated in stage one in order to convert to OSNG. The shifts are derived from a series of matrices which take into account the region of the UK the coordinates lie in. The exact shift is then interpolated from the values obtained from the matrices:

EASTING SHIFTS OBTAINED FROM:

OSGRS80 N		East shift (m)	
1400000	99	104	107
1050000	93	99	106
700000	85	97	108
350000	89	96	105
0	92	96	102
	0	350000	700000
	OSGRS80 E		

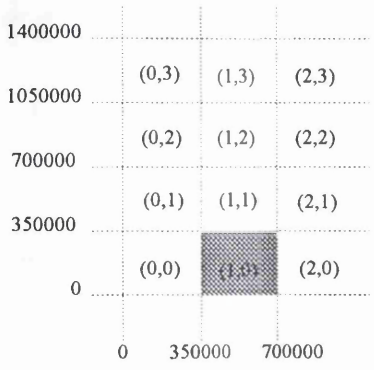
NORTHINGS SHIFTS OBTAINED FROM:

OSGRS80 N		North shift (m)	
1400000	-44	-49	-52
1050000	-47	-52	-54
700000	-58	-62	-62
350000	-75	-75	-78
0	-82	-80	-82
	0	350000	700000
	OSGRS80 E		

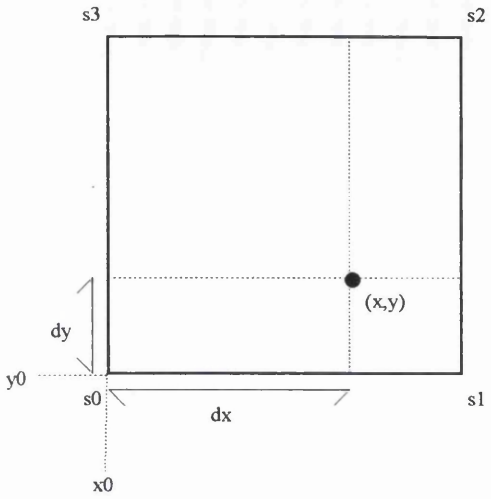
An index is calculated from OSGRS80 Eastings and Northings:

east index = $^{OSGRS80}E/350000$
north index = $^{OSGRS80}N/350000$

The values obtained are rounded down (e.g. 1.9 becomes 1, 0.6 becomes 0). These values are used identify which grid square the coordinates fall into, for example:



If the index is 1,0 then the coordinate will lie inside the highlighted grid square



To determine the exact position of the coordinate in the square, it must be interpolated from the shifts at the four corners of the grid square.

The shifts are gained from the matrices as follows:

OSGRS80 N		East shift (m)	
1400000	99	104	107
1050000	93	99	106
700000	85	97	108
350000	89	**96**	**105**
0	92	**96**	**102**
	0	350000	700000
	OSGRS80 E		

s0 = (1,0) = 96

s1 = (1+1,1) = 102

s2 = (1+1,1+1) = 105

s3 = (1,0+1) = 96

The value of the shift is given in the following formula. This procedure would be repeated for northings shifts by using a second matrix.

Shift = (1-t)(1-u)s0 + t(1-u)s1 + tus2 + (1-t)us3

Where: $t = dx/350000$

$u = dy/350000$

and: $dx = {}^{OSGRS80}E - x0$

$dy = {}^{OSGRS80}N - y0$

Therefore: ${}^{OSNG}E = {}^{OSGRS80}E + Shift$

The process is then repeated with the northing matrix:

$${}^{OSNG}N = {}^{OSGRS80}N + Shift$$

To summarise the transformation involves:

- Convert from WGS84 to OSGR80
- Convert from OSGRS80 to OSNG using shift matrices

Appendix IV Function Hierarchy

The hierarchy illustrated in Figure 5.23 was determined by noting the occurance of the functions and equipment in the *User Functions* section of the User Profile (section 3.2.2). Weights were then allocated to each task based on the importance of the task. The weights were determined based on the findings of the literature review (Chapter 1), and personal experience.

(i) Weighting Applied to Frequency

Task	Weight
Position Determination	5
Orientation	5
Route Planning	3
Route Following	4
Map Reading	5
Time / ETA	2
Others	1

This weight was multiplied with the occurence to give a coefficient, that represented how frequently the function was used, and how important the function was to the User.

(ii) Occurance * Weight = Hierarchy Coefficient

	Required Function(s)								
Task	**Map**	**GPS**	**Compass**	**Map Tools**	**Route Tools**	**Time**	**Data S/R**	**Map Set Up**	**Help**
Position Determination	5	5	0	0	0	0	N/A	N/A	N/A
Orientation	0	0	5	0	0	0	N/A	N/A	N/A
Route Planning	3	3	3	3	3	0	N/A	N/A	N/A
Route Following	4	0	0	4	0	0	N/A	N/A	N/A
Map Reading	5	0	0	5	0	0	N/A	N/A	N/A
Time Estimation	2	0	0	0	2	2	N/A	N/A	N/A
Heirarchy Coefficient	**19**	**8**	**8**	**12**	**5**	**2**	**1**	**1**	**1**

Appendix V GPS Testing Sheet

GPS Testing Sheet				Satellites Used					
Date	*15/08/96*	Static	Mobile	1	2	3	4	5	6
Time	*10:10 am*	Test #	*4*	7	8	9	10	11	12
Weather	*Cloudy*	TTFF*	01:16	13	14	15	16	17	18
Notes:				19	20	21	22	23	24
-2D and 3D fixes obtained				25	26	27	28	29	30
				31	32	33	34	35	36

*TTFF = Time To First Fix

Appendix VI GPS Driver Test Results

(i) Static Test Results

Metres from True Position	Number of Points in Buffer Zone				Total	Total (%)
	Test1	Test2	Test3	Test4		
0-25.5	0	0	29	16	45	22.7
25.5-51	70	1	0	17	88	44.4
51-102	0	64	0	0	64	32.3
103+	0	1	0	0	1	0.5

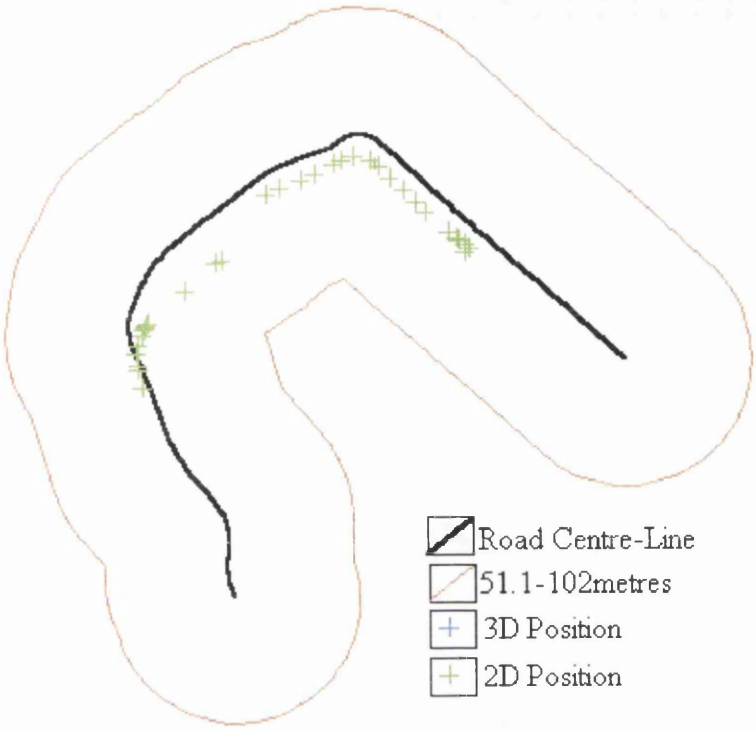
For diagram of static test results see Figure 7.5 p.155

(ii) Mobile Test Results

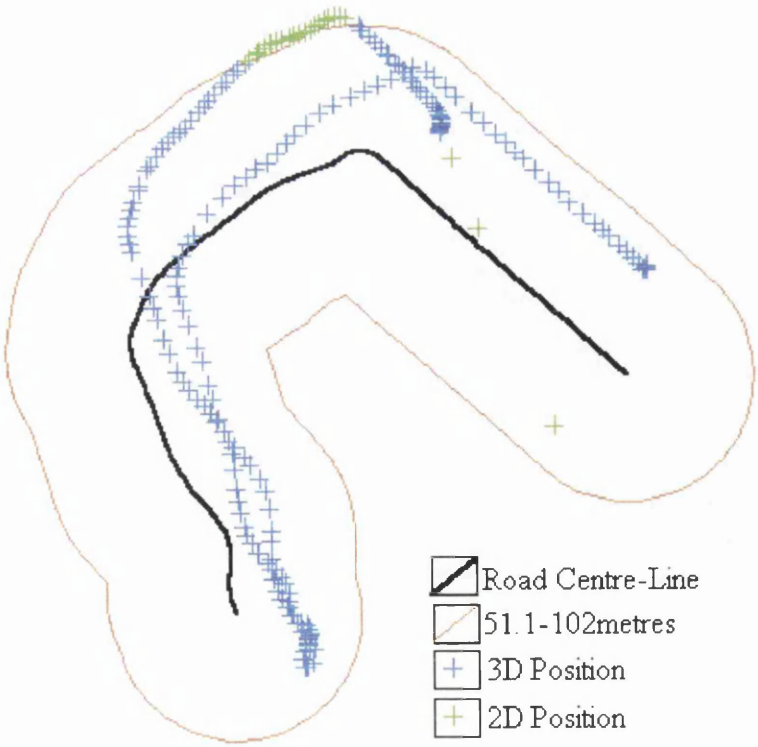
Metres from Road Centre-Line	Number of Points in Buffer Zone				Total	Total (%)
	Test1	Test2	Test3	Test4		
0-102	34	205	198	185	622	96.9
103+	0	19	1	0	20	3.1

Diagrams of mobile test results follow:

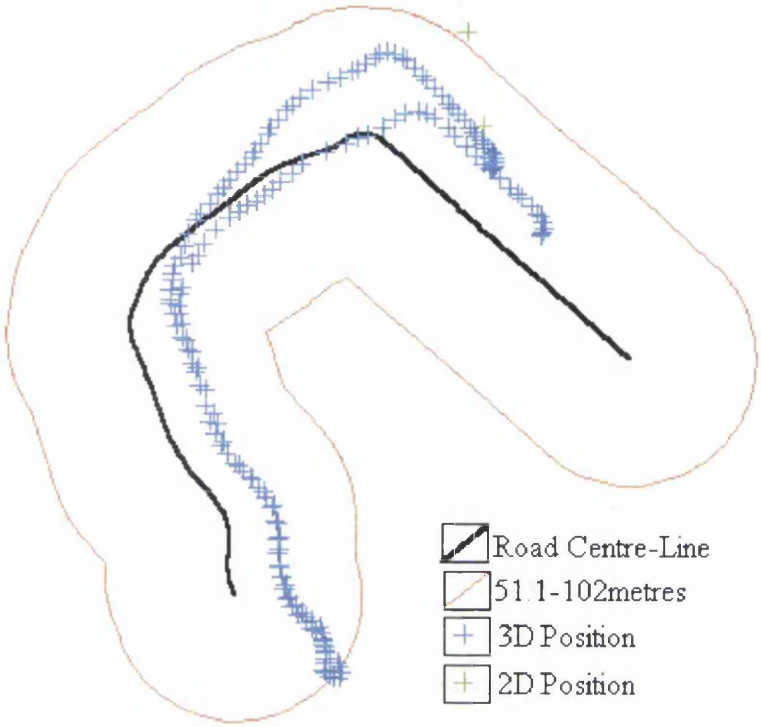
Test 1: Clifton Campus 06-08-96



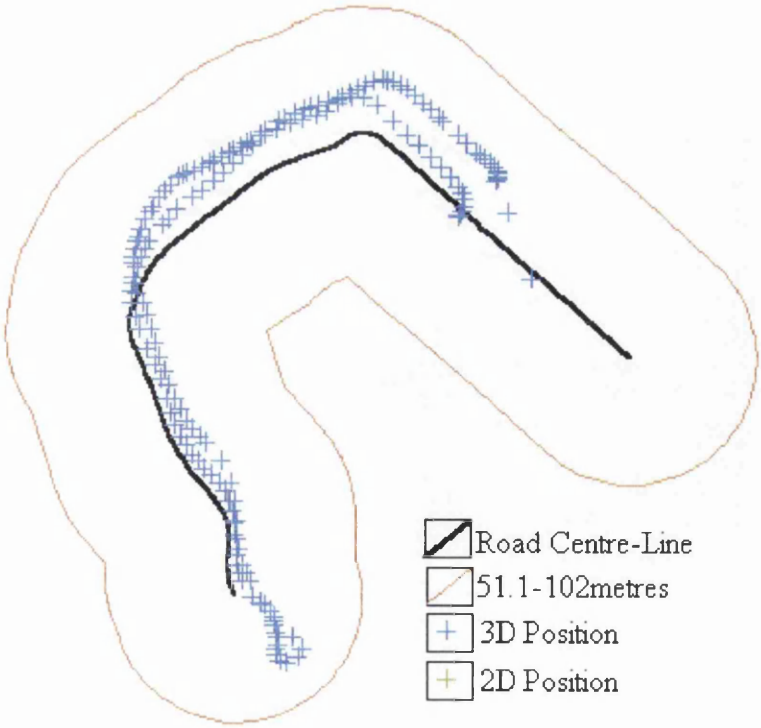
Test 2: Clifton Campus 07-08-96



Test 3: Clifton Campus 12-08-96



Test 4: Clifton Campus 15-08-96



Appendix VII Software User Interface Test Sheet

SOFTWARE USER INTERFACE TEST SHEET

" This is the Electronic Navigation Environment. It allows the User to carry out many navigation tasks including position determination using GPS, orientation using a compass as well as route planning. The user also has complete control over the map in terms of zoom and pan facilities."

Task 1.

ZOOM IN ON AN AREA OF THE MAP

	Interpretation Errors:	**Task Completion Time:**
Cycle 1:	.....	.....
Cycle 2:	.....	.....
Cycle 3:	.....	.....

Task 2.

SELECT THE ROUTE PLANNING ICON AND PLOT A ROUTE ON THE MAP.

	Interpretation Errors:	**Task Completion Time:**
Cycle 1:	.....	.....
Cycle 2:	.....	.....
Cycle 3:	.....	.....

Task 3.

PLOT WAYPOINTS AT THE BEGINNING AND END OF YOUR ROUTE.

	Interpretation Errors:	**Task Completion Time:**
Cycle 1:	.....	.....
Cycle 2:	.....	.....
Cycle 3:	.....	.....

Task 4.

SELECT THE GPS MENU, TURN ON THE GPS, SELECT A TEN SECOND FIX INTERVAL AND SELECT A FIXED MAP DISPLAY.

	Interpretation Errors:	Task Completion Time:
Cycle 1:	.....	.....
Cycle 2:	.....	.....
Cycle 3:	.....	.....

Task 5.

SELECT THE HELP PAGE DESCRIBING THE ZOOM IN TOOL.

	Interpretation Errors:	Task Completion Time:
Cycle 1:	.....	.....
Cycle 2:	.....	.....
Cycle 3:	.....	.....

END OF TEST

TOTAL TIMES:

Cycle 1: .....
Cycle 2: .....
Cycle 3: .....

Appendix VIII Software User Interface Test Results

The following tables show the results gained from the twelve test subjects. All test results are in seconds.

SUBJECT 1	Cycle			
Task	**1**	**2**	**3**	**Mean 2-3**
1	2.1	2.2	1.9	2.1
2	11.0	5.5	6.9	6.2
3	4.0	4.0	4.9	4.5
4	25.1	8.9	8.0	8.5
5	9.0	4.0	4.0	4.0
Total	51.2	24.6	25.7	
Errors	2	0	0	

SUBJECT 2	Cycle			
Task	**1**	**2**	**3**	**Mean 2-3**
1	6.6	1.3	2.0	1.7
2	13.0	4.0	3.0	3.5
3	6.0	3.8	3.8	3.8
4	42.2	8.0	7.0	7.5
5	11.1	4.9	5.0	5.0
Total	78.9	22.0	20.8	
Errors	2	0	0	

SUBJECT 3	Cycle			
Task	**1**	**2**	**3**	**Mean 2-3**
1	31.1	40.0	5.8	22.9
2	29.0	7.0	5.4	6.2
3	7.4	6.7	5.2	6.0
4	51.0	10.7	10.0	10.4
5	6.8	7.3	6.1	6.7
Total	125.3	71.7	32.5	
Errors	6	2	0	

SUBJECT 4	Cycle			
Task	**1**	**2**	**3**	**Mean 2-3**
1	3.7	2.2	1.9	2.1
2	24.5	4.6	4.0	4.3
3	4.8	3.2	3.4	3.3
4	21.5	8.3	8.6	8.5
5	31.0	6.1	4.3	5.2
Total	85.5	24.4	22.2	
Errors	4	0	0	

SUBJECT 5	Cycle			
Task	1	2	3	Mean 2-3
1	29.9	3.5	3.5	3.5
2	16.0	6.3	6.3	6.3
3	5.0	4.7	4.0	4.4
4	24.9	14.0	8.6	11.3
5	9.4	6.3	4.6	5.5
Total	85.2	34.8	27.0	
Errors	3	1	0	

SUBJECT 6	Cycle			
Task	1	2	3	Mean 2-3
1	3.2	2.7	2.2	2.5
2	9.6	6.0	4.0	5.0
3	5.0	5.9	3.5	4.7
4	14.0	9.3	7.5	8.4
5	6.4	4.6	4.7	4.7
Total	38.2	28.5	21.9	
Errors	0	0	0	

SUBJECT 7	Cycle			
Task	1	2	3	Mean 2-3
1	2.5	2.8	2.1	2.45
2	7.8	6.2	4.1	5.15
3	5.6	4.6	4.6	4.6
4	8.9	8.3	10.9	9.6
5	7.6	5.9	5.7	5.8
Total	32.4	27.8	27.4	
Errors	0	0	0	

SUBJECT 8	Cycle			
Task	1	2	3	Mean 2-3
1	8.7	4.8	4.6	4.7
2	9.6	7.2	6.1	6.65
3	9.3	8.7	6.5	7.6
4	12.3	9.8	8.2	9
5	7.6	6.3	6.5	6.4
Total	47.5	36.8	31.9	

SUBJECT 9	Cycle			
Task	1	2	3	Mean 2-3
1	4.8	3.2	3.3	3.25
2	10.2	8.8	7.6	8.2
3	8.2	6.6	6.3	6.45
4	11.1	7.6	7.2	7.4
5	8.2	7.6	6.6	7.1
Total	42.5	33.8	31	
Errors	2	0	0	

SUBJECT 10	Cycle			
Task	1	2	3	Mean 2-3
1	5.9	2.9	2.6	2.75
2	7.9	4.8	4.6	4.7
3	9	6.5	5.8	6.15
4	9.4	7.6	6.9	7.25
5	6.4	5.5	4.9	5.2
Total	38.6	27.3	24.8	
Errors	1	0	0	

SUBJECT 11	Cycle			
Task	1	2	3	Mean 2-3
1	4.7	2.9	2.4	2.65
2	11.1	7.5	7.2	7.35
3	7.8	7.3	6.8	7.05
4	13.6	8.4	7.3	7.85
5	8.8	8.3	7.1	7.7
Total	46	34.4	30.8	
Errors	2	0	0	

SUBJECT 12	Cycle			
Task	1	2	3	Mean 2-3
1	3.3	2.9	2.1	2.5
2	5.1	4.6	4.5	4.55
3	6.9	5.4	4.4	4.9
4	8.3	8.1	6.4	7.25
5	6.5	5.9	5.7	5.8
Total	30.1	26.9	23.1	
Errors	0	0	0	

REFERENCES

Abbott J (1986) Software Testing Techniques NCC Ltd

Ackroyd N, Lorimer R (1990) Global Navigation A GPS User's Guide Lloyds of London Press Ltd

Anderson A (1995) Safer People or Safer Mountains? *RIN'95 Personal Navigation Paper 2, Royal Institute of Navigation*

Antenucci J C, Brown K, Croswell P L, Kevany M J, Archer H (1991) Geographic Information Systems Van Nostrand Reinhold, New York

Archdeacon T (1995) Campfire Tales: Taking GPS on a Hunting Expedition *GPS World* 6(9) p.36-42

AutoDesk Ltd. (1995) DeskMapper *Information leaflet*, AutoDesk Ltd.

Baker R R (1981) Human Navigation and the Sixth Sense Hodder and Stoughton

Bannister A, Raymond S, & Baker R (1992) Surveying Longman

Bergum BO, Bergum JE (1981) Population Stereotypes: An attempt to measure and define. *Proceeding of the Human Factors Society, Human Factors Society* p.622-665

Bodker S (1991) Through the Interface Lawrence Erlbaum Associates, New Jersey

Breach M (1993) Engineering Surveying The Nottingham Trent University

British Computer Society (1995) A Glossary of Computing Terms 8th Edition Longman

Brown AR, Sampson WA (1973) Program Debugging MacDonald

Bugayevskiy LM, Snyder JP (1995) Map Projections Taylor and Francis

Cassel D (1983) The Structured Alternative: Program Design, Style and Debugging Prentice-Hall

Cross PA, Hollwey JR, Small LG (1989) Working Paper No.2: Geodetic Appreciation University of East London

Defense Mapping Agency (1991) Department of Defense World Geodetic System (1984) Defense Mapping Agency, Fairfax Virginia

Diaper D, Winder R (1987) People and Computers III Cambridge University Press

Dijkstra EW (1965) Programming Considered as a Human Activity *Proceeding of the (1965) IFIP Congress* p.213-217

Dodson AH (1995) The Shape of the Earth *NAV '95 Mapping and the Display of Mapping Information,* Paper **4**, *Royal Institute of Navigation*

Ewing C E Mitchell M M (1970) Introduction to Geodesy Elsevier, New York

Fagan ME (1986) Advances in Software Inspection *IEEE Transactions on Software Engineeering* **SE-12**(7)

Fowler SL, Stanwick VR (1995) The GUI Style Guide AP Professional

Gilbert C (1996a) Decoding Advertisers Claims for GPS Accuracy *Mapping Awareness* **10**(4) p.40-43

Gilbert C (1996b) The Software with your GPS *Mapping Awareness* **10**(8) p.32-34

Gilbert C (1996c) The end of Selective Availability: a signal difference, or just a different signal? *Mapping Awareness* **10**(6) p.30-32

Gilbert C (1995) Could do Better ? *Mapping Awareness* **9**(3) p.37-39

Gurewich N. Gurewich O. (1995) Visual Basic v4.0 SAMS, Indianapolis, Indiana

Hammond R McCullagh P S (1989) Quantitive Techniques in Geography 2nd Edition Oxford University Press

Harley J B (1975) Ordnance Survey Maps: A Descriptive Manual HMSO

Hartley S (1996) Ordnance Survey Digital Mapping for Land Navigation *Journal of Navigation* **49**(2) p.163-168

Helander M (1992) Handbook of Human Computer Interaction North-Holland Publishing

Horton W (1994) The Icon Book: Visual Symbols for Computer Systems and Documentation John Wiley and Sons. Inc.

Kaner C (1988) Testing Computer Software TAB, Blue Ridge Summit, Virginia

Keates J S (1989) Cartographic Design and Production 2nd Edition Longman S & T

Keay W (1989) Land Navigation HMSO

Kennie T J M, Petrie G P (eds.) (1993) Engineering Surveying Technology Paperback Edition Blackie A & P

King D (1988) Creating Effective Software Prentice-Hall

King RW,Masters EG, Rizos C, Stoltz A, Collins J (1987) Surveying with Global Positioning System Dummlers Verlag, Bonn

Kobara S (1991) Visual Design with OSF/Motif Addison-Wesley, Sydney

Kumar M (1993) World Geodetic System 1984: A Reference Frame for Global Mapping, Charting and Geodetic Applications *Surveying and Land Information Systems* **53**(1) p.53-56

LaBudde K (1987) Structured Programming Concepts McGraw-Hill

Lawrence GRP (1971) Cartographic Methods Methuen

Leica AG (1993) User Manual SKI - Static Kinematic Software Leica AG, Heerbrugg, Switzerland

Leick A (1990) GPS Satellite Surveying John Wiley & Sons. Inc.

Maling DH (1992) Coordinate Systems and Map Projections 2nd Edition Pergamon Press

Mapinfo Corporation (1994a) Mapinfo User's Guide Mapinfo Corporation, New York

Mapinfo Corporation (1994b) MapBasic Reference Mapinfo Corporation, New York

Mapinfo Corporation (1994c) MapBasic User's Guide Mapinfo Corporation, New York

Mapinfo Corporation (1995) Mapinfo v3.0 for Windows *Information leaflet*, Mapinfo Corporation, New York

Mayhew D J (1992) Software User Interface Design Prentice-Hall Inc.

McGowan L, Kelly JR (1975) Top-Down Structured Programming Techniques 1st Edition Petrocelli / Charter, New York

Methley BDF (1991) Geometrical Geodesy *Department of Topographic Science, University of Glasgow Unpublished Lecture Notes*

Mikhail E M Gracie G (1981) Analysis and Adjustment of Survey Measurements Van Nostrad Reinhold, New York

Moore D T (1995) Hand Held GPS Surveying *RIN'95 Personal Navigation,* Paper **5**, *Royal Institute of Navigation*

Nordby K, Helmersen PH, Gilmore DJ, Arnesen SA (eds.) (1995) Human-Computer Interaction: Interact '95 1st Edition Chapman & Hall

Ordnance Survey (1983) Transverse Mercator Projection: Constants, Formulae and Methods HMSO

Ordnance Survey (1995a) National Grid / ETRF89 Transformation Parameters Ordnance Survey

Ordnance Survey (1995b) The Ellipsoid And Transverse Mercator Projection Ordnance Survey

Ordnance Survey (1996) Digital Map Data and Customised Services Ordnance Survey

Owen T, Pilben E (1992) Ordnance Survey - Mapmakers to Britain Since 1791 HMSO

Parent PJ (1988) Geographic Information Systems: Evolution, academic involvement and issues arising from the proliferation of information *Masters Thesis, University of California, Santa Barbra*

Press F, Siever R (1986) Earth 4th Edition, WH Freeman, New York

Pritchard P (1988) Introduction to Programming using Macintosh Pascal Addison-Wesley, Sydney

Raisz E (1948) General Cartography MacGraw-Hill

Reeve D (1996) ArcView 2.1, Atlas GIS 3.01 and Mapinfo 3.02 *Mapping Awareness* **10**(2) p.42-43

Robinson A H Randall D S Morrison J L Muehrcke P C (1985) Elements of Cartography 5th Edition, John Wiley & Sons

Rockwell (1996) NavCard LP *Information leaflet*, Newport Beach, California

Rubin T (1988) User Interface Design for Computer Systems John Wiley & Sons

Seeber G (1993) Satellite Geodesy de Gruyter, Berlin

Shneiderman B (1987) Designing the User Interface: Strategies for Effective Human-Computer Interaction Addison-Wesley, Sydney

Silva (1994a) GPSMON - Interface to the GPS Compass Silva Production AB, Kuskvagen, Sweden

Silva (1994b) Silva Bus Specification Version 1.19 Silva Production AB, Kuskvagen, Sweden

Starbuck JA (1995) Military Needs for Personal Navigation *Journal of Navigation* **48**(1) p.77-80

Strategic Mapping Inc. (1995) Atlas GIS v3.0 for Windows *Information leaflet*, Strategic Mapping Inc., Santa Clara, California

Sutcliffe A G (1995) Human-Computer Interface Design 2nd Edition, MacMillan

Sutcliffe AG (1988) Human-Computer Interface Design 1st Edition, MacMillan Education Ltd

Swann T (1995) Mastering Borland C++ 4.5) SAMS, Indianapolis, Indiana

Telecom Design Communications Ltd. (1995) TDC GPS Seminar *TDC GPS Seminar Proceedings*

Tippet J (1995) Walkers Navigational Needs *RIN'95 Personal Navigation* Paper 7, *Royal Institute of Navigation*

Travis D (1991) Effective Color Displays: Theory and Practice EACE

Trimble Navigation Ltd. (1994a) SVeeSix System Designer's Reference Manual Trimble Navigation Ltd., Sunny Vale, California

Trimble Navigation Ltd. (1994b) Mobile GPS Programmer's Guide Trimble Navigation Ltd., Sunny Vale, California

Vanicek P Krakiwsky E (1986) Geodesy: The Concepts 2nd Edition, Elsevier, New York

Wale T (1995) Mapping and the Display of Information *NAV '95 Mapping and the Display of Mapping Information,* Paper **29**, *Royal Institute of Navigation*

Walker K (1986) Mountain Navigation Techniques Constable

Walker S (1995) The Course of Navigation *Navigation News, March / April,* p.14-16

Wright E (1995) Summer Fun: GPS Takes a Vacation *GPS World* **6**(6) p.22-30

Yourdon E (1975) Techniques of Program Structure and Design Prentice-Hall Inc.