University of Southern Queensland
Faculty of Engineering and Surveying

Centreline Tools Upgrade

A dissertation submitted by

Robert William Thompson

in fulfilment of the requirements for

Courses ENG4111 and 4112 Research Project

towards the degree of

Graduate Diploma of Geomatic Studies (Geographic Information Systems)

Submitted: October, 2008

Abstract

The state-controlled road network is Queensland's single largest built community asset and the Department of Main Roads is responsible for its operation and the delivery of projects for its enhancement and maintenance. The majority of road related asset data is stored in linear referenced format, which cannot be represented spatially in a MapInfo GIS environment without translation.

Centreline Tools is a MapBasic utility developed by Main Roads to translate between linear and spatial forms of data. Centreline Tools performs dynamic segmentation operations not supported by native MapInfo, and enables linear referenced road asset data to be viewed and analysed in MapInfo with coordinated-based vector data.

This dissertation describes the development of an upgraded version of the Centreline Tools utility. A mixture of existing and new ideas was incorporated into the development process while the original look and feel were retained for operational consistency. Several long standing issues with its operation were resolved and reporting has been improved. Modifications to the organisation and structure of the upgraded version have resulted in a more maintainable product. Early evidence suggests that the majority of the benefits expected from the project have been realised.

An iterative implementation of the System Development Life Cycle methodology was adopted for the project and proved to be very effective. This strategy facilitated extended testing of the logic and procedures at the core of the upgrade and allowed feedback to be evaluated and integrated into subsequent development releases.

The Centreline Tools upgrade delivered by this project has now been released into production.

University of Southern Queensland

Faculty of Engineering and Surveying

ENG4111 & ENG4112 *Research Project*

Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled "Research Project" is to contribute to the overall education within the student's chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof Frank Bullen

Dean

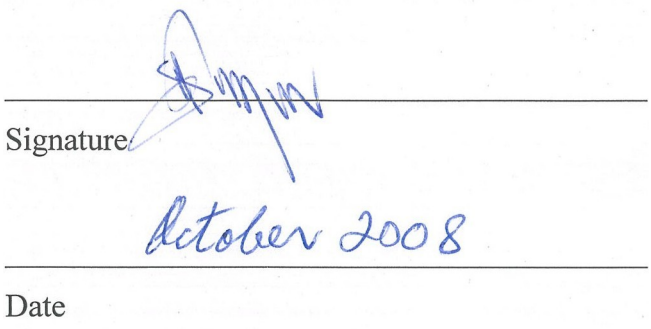
Faculty of Engineering and Surveying

Certification

I certify that the ideas, designs and experimental work, results, analyses and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Robert William Thompson
Student Number: 0031210039



Acknowledgments

I would like to take this opportunity to thank my project supervisor, Mrs Xiaoye Liu, University of Southern Queensland, for her advice and support throughout this project.

I would also like to thank the following people for the valuable feedback they provided and acknowledge their contribution to the project:

- Alex Appleman, Department of Main Roads, Townsville.
- Debra Barney, Department of Main Roads, Bundaberg.
- Maxine Maguire, Department of Main Roads, Townsville.
- Stephen Clague, Department of Main Roads, Mackay.
- Tony Martin, Department of Main Roads, Cairns.

Finally, I would like to thank the staff of Geospatial Technologies Branch, Department of Main Roads for their continued support during the project, in particular:

- Tony Bitz, Director (Geospatial Technologies), and
- Richard Jeffries, Senior Advisor (GIS).

Table of Contents

List of Figures

List of Tables

Glossary

ARMIS	A Road Management Information System
Chainage	Driven distance along a road from its start point
GIS	Geographic Information System
GIS&T	Geographic Information Science & Technology
GPS	Global Positioning System
Line	A MapInfo vector object with two nodes, one at each end
Link	Component of a MapInfo vector polyline between two nodes
LRS	Linear Referencing System
MapBasic	MapBasic Version 8.0
MapInfo	MapInfo Professional Version 8.0/9.01
MR	Queensland Department of Main Roads
Node	The end point of a link or line
Point	Spatial object represented by a x, y coordinate pair
Polyline	A connected sequence of links that is not closed
Portion	Part of a segment object occurring within a road section
Reference Point	Point of known location on the road network
REFLEN	Attribute for driven length of a road segment
REFOFFSET	Attribute for chainage at the start of a road segment
RII	Road Information Integrator
RIP	Roads Implementation Program
RP	Reference Point
RRS	Road Reference System
SAS	Spatial Application Suite
SCALEFACTOR	Attribute for conversion between object length and chainage
SDLC	System Development Life Cycle

Section	Part of a particular road between two liner references or points
Segment	Part of a particular road between two RPs
SQL	Structured Query Language

CHAPTER 1
INTRODUCTION

1.1 Background

The state-controlled road network consists of approximately 33,500 kilometres of public roads and 2,838 bridges, and is Queensland's single largest built community asset (Department of Main Roads 2007). Responsibility for the operation of the road network and the delivery of projects for its enhancement and maintenance lies with the Department of Main Roads. In order to meet this obligation, Main Roads requires the capacity to store and manipulate large volumes of road network related data.

Historically, Main Roads has stored road asset data in departmental databases in linear referenced format, in which the location of an entity or event is described as a measurement along a specified road. However, linear references are textual descriptions and cannot generally be represented spatially in a Geographic Information System (GIS) without some form of translation. Such translations require a conversion between linear reference descriptions and geospatial coordinates and involve a process known as dynamic segmentation.

Geographic Information Science and Technology (GIS&T) is widely used in Main Roads, most commonly in a MapInfo Professional desktop environment. In addition to out-of-the-box MapInfo, the department utilises a suite of custom-built tools and utilities developed in-house in MapBasic, MapInfo's development language. This suite of tools, referred to internally as the Spatial Application Suite (SAS), enhances and extends the spatial capability of MapInfo in the department and supports a number of

business operations and processes. One important member of this suite performs a number of operations based on dynamic segmentation, which are not available in native MapInfo Professional. This utility is known as Centreline Tools.

The dynamic segmentation capabilities of Centreline Tools enable linear referenced road asset data to be viewed and manipulated in a GIS environment with regular spatial data such as the road centreline, cadastre, aerial photography and satellite imagery.

Activity logging implemented to monitor usage of the SAS indicates that there are approximately one thousand (1000) users of MapInfo Professional within Main Roads. Activity logs also indicate that many of these users load Centreline Tools on a regular basis, suggesting that it performs an important role in supporting Main Roads business operations.

Centreline Tools was originally released for use in Main Roads circa 1998. Since that time it has received very little maintenance, contributing to an apparent gradual degradation of operational reliability and performance in recent years. In order to ensure the continued availability of Centreline Tools, the issues impacting on its declining reliability needed to be addressed.

1.2 Aims and Objectives

The aim of this project was to upgrade the Centreline Tools utility.

Specifically, the objectives of the project were to upgrade the most regularly used utility operations, which involved:

- determining the linear reference of a specified coordinate based location,
- finding the coordinate based location of a specified linear reference,
- creating coordinate based spatial objects representing the linear references contained in a specified MapInfo input table, and
- determining the linear references representing the point objects contained in a specified MapInfo input table.

Supporting functionality enabling the display and manipulation of data generated by these operations was also to be upgraded.

Additional objectives, while desirable, were not initially considered to be possible within the time and resource constraints of this project, but would be considered if time permitted. These involved:

- upgrading the remaining operations that generate spatial objects representing sections of the road network defined by specified start and end points or linear references, and
- improving the reporting capabilities of the utility in relation to the outcome of processes performed on MapInfo tables.

1.3 Expected Benefits

It was expected that an upgrade of Centreline Tools would deliver a number of operational and organisational benefits to the department.

In an operational context, benefits might include:

- the correction of some long standing issues with the utility's operation,
- improved reliability, consistency and dependability,
- simpler to use, less ambiguous user interface dialogs,
- improved reporting capabilities, and
- increased flexibility in linear referenced data format selection.

It was hoped that the delivery of these benefits would enhance useability and improve user confidence in the utility. As well as benefiting regular users, flow-on effects might include an increase in the number of district officers prepared to utilise dynamic segmentation as a means to display and manipulate linear referenced data in a GIS environment.

From an organisational perspective, it was expected that an upgraded Centreline Tools would be easier to maintain, saving time and resources for the department. It would also make the development of enhancements and extensions a more feasible proposition.

1.4 Dissertation Overview

Chapter 1, this chapter, provides some background information and introduces the aims and objectives of the project, together with a summary of the expected benefits.

Chapter 2 reviews the concepts and technologies that underpin dynamic segmentation. The relationship between Centreline Tools and the vector-based representation of the state-controlled road network known as '*sc_roads*' is described.

Chapter 3 outlines the methodology adopted for the project and describes the design strategies employed for the implementation of dynamic segmentation.

Chapter 4 covers the development phase of the project and includes a description of the strategies used for testing and significant coding issues encountered.

Chapter 5 presents a detailed description of the operation and behaviour of the upgraded version of Centreline Tools.

Chapter 6 offers some concluding remarks together with an assessment of the benefits derived from the project and the potential for further work.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction

This chapter reviews the concepts and technologies that underpin the project.

Spatial and linear referencing systems are introduced, together with the origin of, and continuing requirement for dynamic segmentation.

The chapter goes on to describe the structure of the state-controlled road network and how it is represented as a road centreline using the vector data model in MapInfo. The specific characteristics of the road centreline that support the implementation of dynamic segmentation in Centreline Tools are discussed.

Finally, the terminology used throughout this dissertation to describe particular components of vector structures is explained to avoid any confusion with common usage.

2.2 Spatial Referencing

Geographic Information Systems (GIS) commonly use the vector data model to represent geographic features. In the vector data model, two-dimensional x, y coordinates are used to store the shape of spatial entities (Korte 2001). That is, a point feature such as a road sign or light pole is represented by a single x and y coordinate

pair. A linear feature such as a road or river is represented by a continuous string of x and y pairs (Davis 1996). The vector data model works well for features with static boundaries (ESRI 2002).

Using coordinate systems to locate features on the surface of the earth in this way is known as spatial referencing (Kothuri, Godfrind & Beinat 2004).

2.3 Linear Referencing

Spatial coordinates are not the only way to locate objects. Data related to linear features such as roads are often collected and referred to on the basis of a relative position on the road. That is, a location on a linear feature can be described by a measure value such as distance, with respect to some known point on the feature such as its start point. This type of location referencing using a measure is called a Linear Referencing System (Kothuri, Godfrind & Beinat 2004) (LRS). For example, the description '55 kilometres along road 12A' uniquely describes a location in space without the use of geographic coordinates. This is a very efficient way of storing information related to linear structures such as road networks. An interesting aspect of linear referencing is that a two-dimensional linear network is reduced to a one-dimensional linear list (Longley et al. 2005). Linear referencing therefore underpins an important model for the collection and storage of information by agencies responsible for the management of linear networks.

The Queensland Department of Main Roads employs a specific implementation of linear referencing known as the Road Reference System (RRS) as a basis for the storage and management of road asset information (Department of Main Roads, unpub.). In the Road Reference System, a system of points of known location called Reference Points (RPs) enable any part of the road network to be described in terms of a road, carriageway and distance measurement. The majority of road asset data is stored in this format in a departmental database known as ARMIS (A Road Management Information System).

The use of linear referencing for the collection and storage of information related to the road network and road corridor is entirely appropriate for linear assets. However,

analysing, visualising, displaying and mapping these data in association with other contextual spatially referenced data can be problematic.

2.4 Dynamic Segmentation

Dynamic segmentation was developed out of a need to geographically represent features whose locations are recorded as a relative distance along a linear feature (ESRI 2002). ESRI (2002) goes on to describe dynamic segmentation as the process of displaying linearly referenced data in a geographic environment. Kothuri, Godfrind & Beinat (2004) describe dynamic segmentation as the ability to generate points or line segments dynamically from measure information.

Linear referencing in association with dynamic segmentation has some advantages over spatial referencing. Small portions along a road can have attributes attached dynamically without impacting on the base feature. That is, portions and attributes can be derived as needed at run-time (Wood 2000). In addition, this method enables multiple sets of attributes to be associated with any portion of an existing linear feature, independent of its beginning and end. These attributes can be displayed, queried, edited and analysed without affecting the underlying linear feature's geometry (ESRI 2002).

Some GIS software packages and environments offer dynamic segmentation capabilities as a standard feature. For example, GeoMedia (Intergraph Corporation 2001), ArcGIS (ESRI 2001, 2002), and Oracle Spatial (Kothuri, Godfrind & Beinat 2004) each provides an implementation of dynamic segmentation using their own proprietary data structures and function sets. However, the MapInfo Professional software used in Main Roads has no such capability in its native form. An implementation of dynamic segmentation is therefore required to spatially represent linear referenced road asset information in the MapInfo environment.

2.5 Dynamic Segmentation in Centreline Tools

In recognition of the importance of spatially representing linear referenced data, Main Roads has developed a number of implementations of dynamic segmentation in recent years.

Centreline Tools was one such application, designed to operate within the MapInfo Professional environment. Centreline Tools, was developed in MapInfo's development language, MapBasic and was released for use within Main Roads circa 1998.

The original version of Centreline Tools performed a number of operations based on dynamic segmentation (Department of Main Roads 1999). Centreline Tools enabled direct translation between spatial coordinates and linear references in a simple to use interactive environment. Centreline Tools also supported dynamic segmentation processing of input tables containing linear references, as well as determining the linear references for input tables of point objects. Appendix K contains the user guide for the operation of the original version of Centre Line Tools.

The implementation of dynamic segmentation in Centreline Tools was built around the properties of a specifically structured MapInfo table called *sc_roads,* which is a vector-based representation of the state-controlled road network.

The road network is divided into segments based on physical and structural elements such as intersections, carriageways, overpasses, underpasses and bridges. The *sc_roads* table contains a vector-based graphic object representing each segment of the road network. The graphic objects of *sc_roads* therefore collectively represent the entire state-controlled road network in its correct geospatial location. Figure 2.1 illustrates the *sc_roads* representation of the state-controlled road network in a MapInfo environment.

Each segment of the road network is also described by the Road Reference System (RRS). In the RRS, a road segment is a section of road occurring between two Reference Points (RPs) (Department of Main Roads, unpub.). The RRS therefore provides the linear reference at the start and end point of each road segment in the road network in the form of road identifier, carriageway type and distance from the start of the road. By including these linear reference elements in the attribute set of the

sc_roads table, a direct relationship is created between each vector object, and the RRS. The attribute set from the *sc_roads* table is displayed in Figure 2.2.

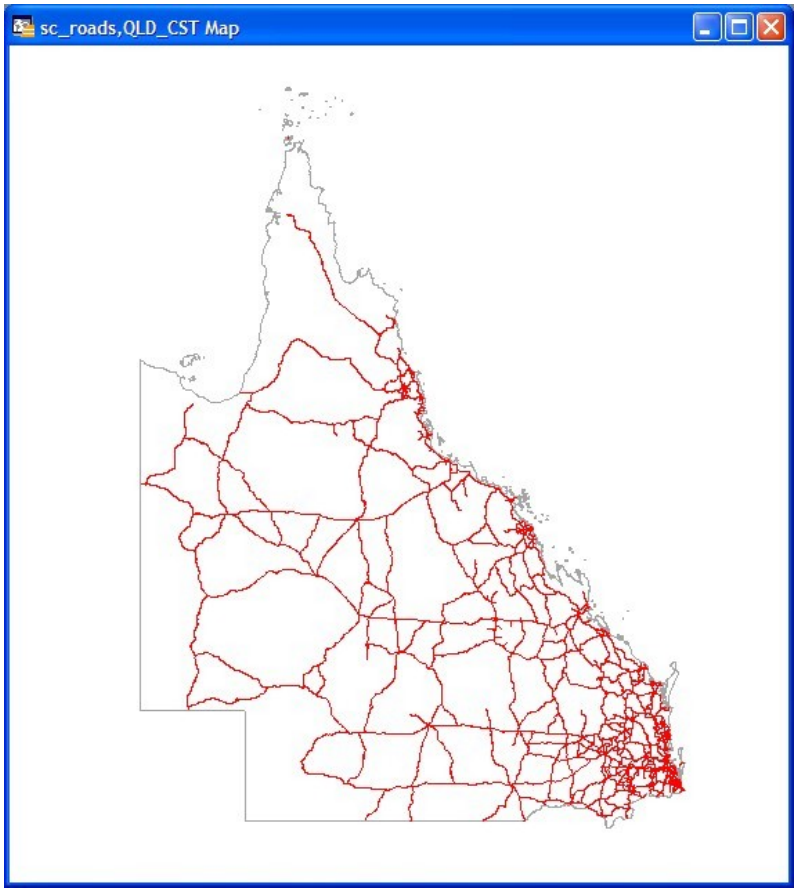


Figure 2.1 – Spatial Objects of *sc_roads*

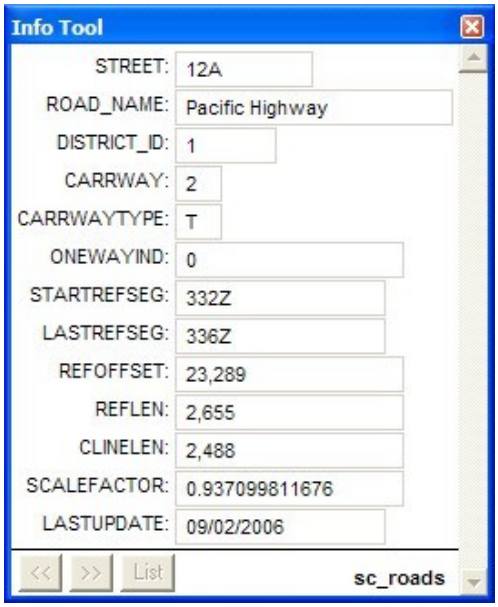


Figure 2.2 – Attributes of *sc_roads*

It is this relationship that underpins the implementation of dynamic segmentation in Centreline Tools.

A number of the attributes of the *sc_roads* table contain information necessary for the dynamic segmentation process. These attributes are listed in Table 2.1 and their specific characteristics are described below.

Attribute Name	Attribute Description
STREET	Road Identifier for the segment
CARRWAY	Carriageway Code (infers segment direction)
REFOFFSET	Chainage at the start of the segment in metres
REFLEN	Driven length of the segment in metres
CLINELEN	Vector object length in metres
SCALEFACTOR	Measurement correction factor for the segment

Table 2.1 – Attributes of *sc_roads* used in Dynamic Segmentation

STREET

In the RRS, an alphanumeric code is assigned to each individual road in the road network. This code value is contained in the STREET attribute of *sc_roads* and identifies the specific road to which the segment object belongs.

CARRWAY

Every road has an assigned direction, which is specified when the road is legally gazetted. The gazetted direction of a road is the direction in which the RRS linear reference measurement value, known as 'chainage' or 'through distance' increases. Gazettal direction is consistent over an entire road.

Because of the complex structure of the road network, some carriageway types have a direction that is opposite to, or against gazettal direction. In these carriageway types, the direction of the segment object matches the direction of the carriageway. Therefore, the RRS chainage increases in a direction opposite to the direction of the segment object. A diagrammatic representation of carriageway types and their relationship with gazettal

direction is presented in Figure 2.3. The *sc_roads* table represents the road network to the carriageway level only and does not include lanes.

The *sc_roads* attribute CARRWAY contains the code value representing the carriageway type of the specific road segment. This attribute therefore also represents the direction of the road segment in relation to the gazettal direction of the road to which the segment belongs.

REFOFFSET

The REFOFFSET attribute of *sc_roads* contains the RRS linear measure or chainage value at the starting point of the road segment. Since chainage increases with gazettal, the start chainage of the segment will occur at the start or end point of the segment object, in object direction, depending on the carriageway type of the segment.

REFLEN

The REFLEN attribute contains the actual three-dimensional length of the road segment according to the RRS. This value is known as the driven length of the segment.

CLINELEN

The two-dimensional length of the vector object representing the road segment is contained in the CLINELEN attribute.

SCALEFACTOR

An obvious disparity exists between the three-dimensional driven length of a road segment relative to the RRS (REFLEN), and the two-dimensional length of the MapInfo vector object representing it (CLINELEN). The SCALEFACTOR attribute contains a correction factor for this disparity. A mathematical relationship therefore exists between these three attribute values such that:

SCALEFACTOR = *CLINELEN* ÷ *REFLEN*, and

REFLEN = *CLINELEN* ÷ *SCALEFACTOR*.

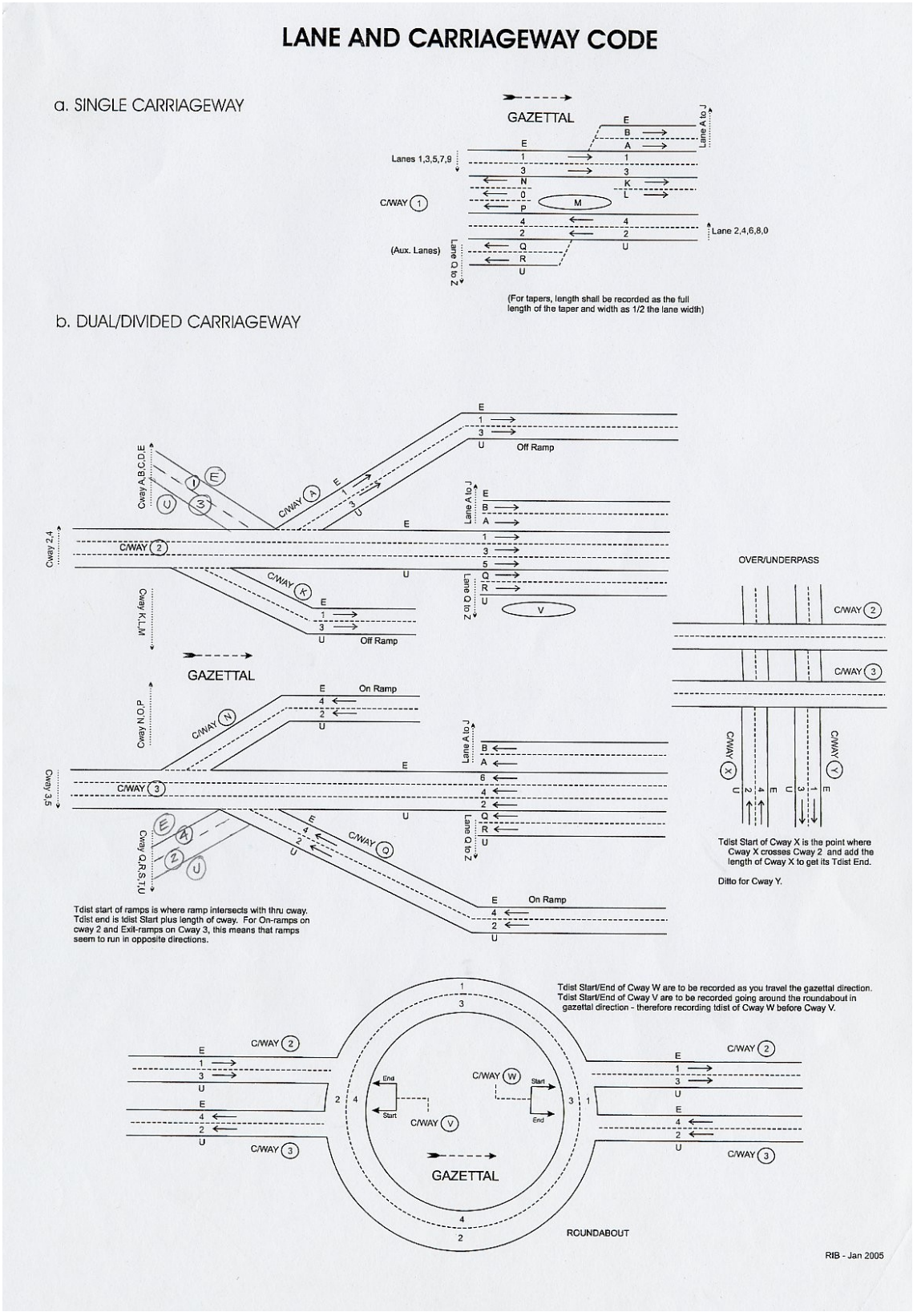


Figure 2.3 – Road Structure and Direction

Source: Roads Information Branch, Department of Main Roads, 2005.

2.6 Terminology

Dynamic segmentation necessarily involves the processing and manipulation of the fundamental graphic objects that comprise the road centreline. As is typical of GIS software developers, MapInfo utilises a specific implementation of vector structures with its own terminology to describe graphic objects and their component elements. MapInfo Corporation (2005b) provides the following definitions of vector structure elements:

- A line is an object with two nodes, one at each end.
- A polyline is a connected sequence of lines that are not closed.
- A node is an end-point of a line object or an end-point of a line segment that is part of a polyline.
- A line segment is a portion of a polyline between two nodes.

It should be noted that MapInfo does not use the term 'vertex', commonly used in other texts such as Davis (1996), to describe the non-end points of a polyline.

MapInfo uses the term 'line segment' to describe the straight component of a polyline between two nodes. However, the RRS refers to a 'segment' as that portion of the road network that lies between two Reference Points. In order to avoid confusion, therefore, for the purposes of this project the term 'segment' is used to describe the spatial object representing one record in the road centreline MapInfo table *sc_roads*. The term 'link' is substituted for 'line segment' and refers to the straight component of a polyline between two nodes.

Figure 2.4 illustrates the structure of the polyline vector objects used in the MapInfo table *sc_roads*. In this example, *sc_roads* has been thematically mapped by carriageway type to enhance the illustration. Nodes and object direction are displayed and carriageways are labelled.

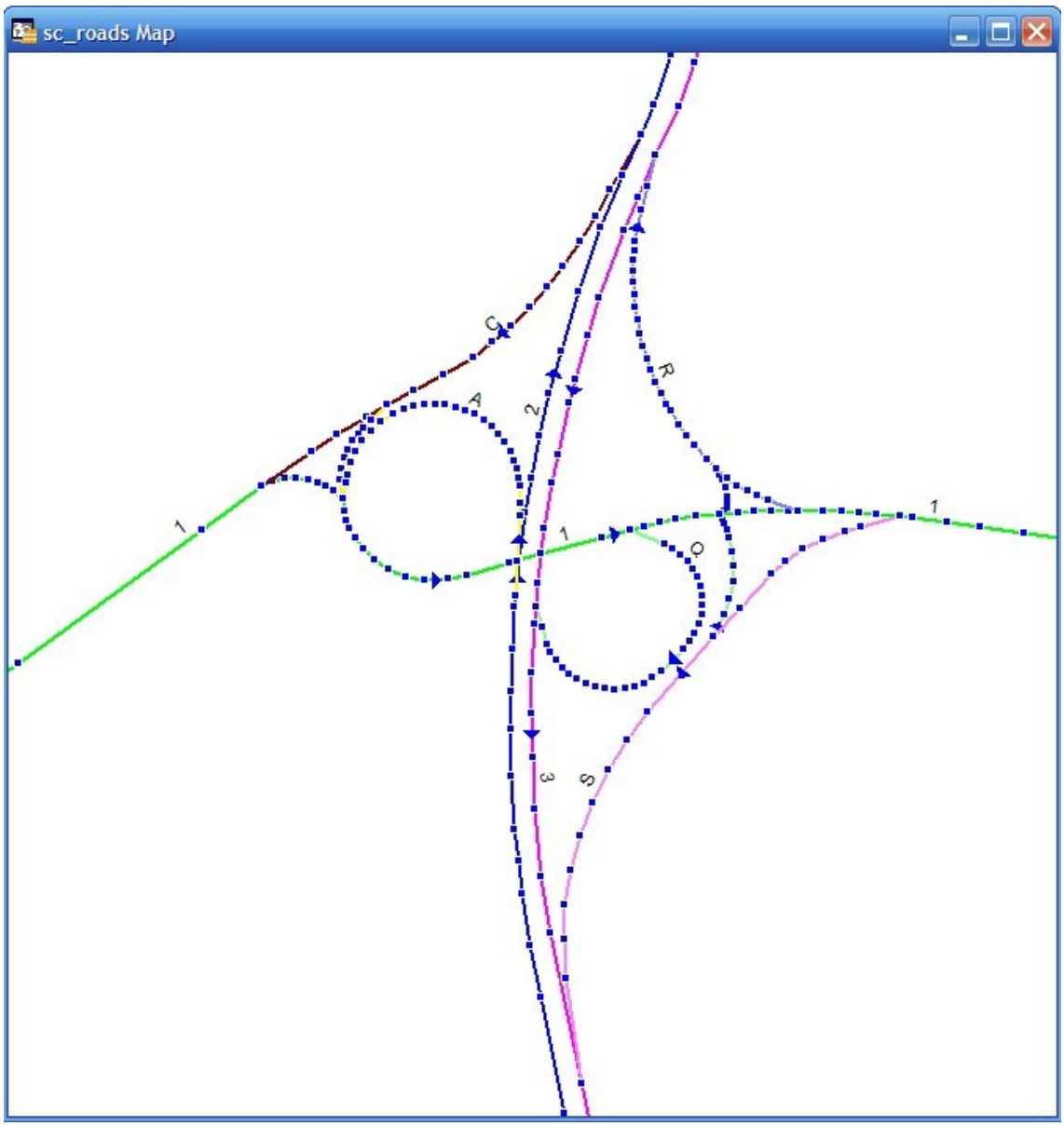


Figure 2.4 – Example *sc_roads* Polylines

CHAPTER 3
METHODOLOGY

3.1 Overview

The methodology followed for this project is known as the System Development Life Cycle (SDLC). The SDLC is a well-established methodology developed for the analysis and design of information systems (Senn 1989; Scott et al. 1996) and involves a set of distinct phases (Scott et al. 1996). Figure 3.1 shows the various phases of the System Development Life Cycle as described by Scott et al. (1996).

An iterative implementation of the SDLC, as described by Apan (2002, p. 2.5), was chosen as the most appropriate methodology for the upgrade of Centreline Tools.

Initially, the interactive operations involving single location translations between spatial coordinates and linear references were analysed, designed, developed and distributed to selected officers for testing. While this development version was being tested and evaluated, the remaining operations, which build on these fundamental processes, were analysed, designed and developed.

This agile development approach was made possible by the familiarity of the testing officers with the existing utility and its functionality, and offered the following advantages:

- The logic and procedures at the core of Centreline Tools were under test for an extended period.
- Comprehensive testing in a true operational environment was enable.

- Experienced users with extensive local knowledge of the road network were likely to recognise erroneous calculations.
- Users had access to the basic operations in the new version soon after they were developed.
- Feedback from regular users of the utility would be available for consideration at a much earlier stage in the development life cycle.
- The completed upgrade would be available more quickly overall.

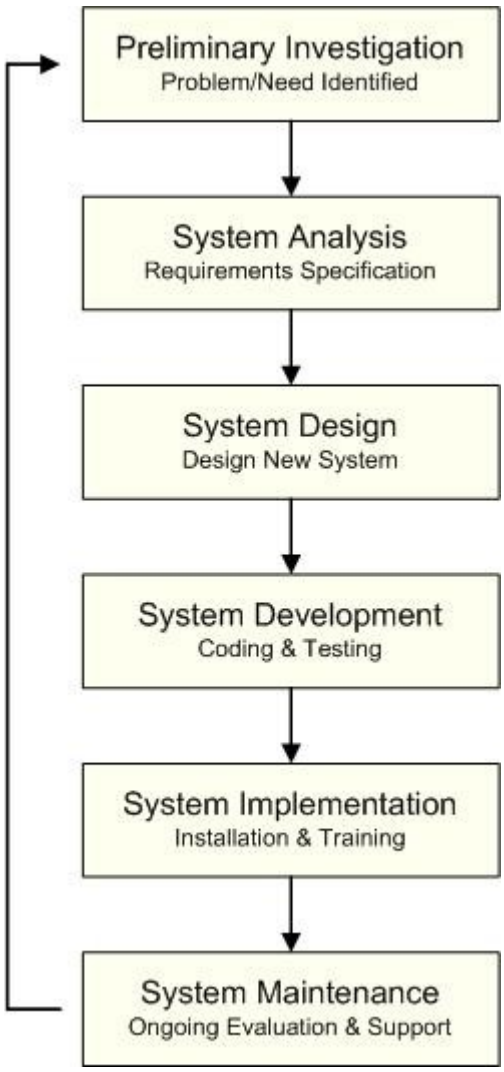


Figure 3.1 – System Development Life Cycle

The main disadvantage of this approach was that some procedures might not be as well planned and functionally independent as possible in the initial iteration of the

development cycle. This would result in some rework to maximise their useability and flexibility in subsequent iterations of the SDLC.

The following sections describe the steps taken in each phase of the SDLC.

3.2 Preliminary Investigation

As an active operational system, Centreline Tools has been the subject of periodic reviews to assess its functional suitability and operational effectiveness. As a result of these reviews, a number of issues with the existing version have been known to exist for some time. The need for maintenance of the utility was therefore recognised and well overdue, and led the proposal of this project.

The most obvious problem occurred in determining the linear reference at a specified spatial location, which is the most regularly used utility operation. An incorrect chainage was usually generated on the first occasion the operation was performed. It appears that the utility calculated a chainage based on the commencement of a road segment, rather than from the start of the road. While regular users of the utility were aware of this error and worked around it, the continuation of this behaviour was clearly unacceptable.

An operational issue was identified in relation to the input of a linear reference to be translated to spatial coordinates (Martin, AM 2008, pers. comm.). A list of all available road identifiers was generated every time the operation was performed. This process demanded additional interactive selections and slowed the operation unnecessarily for regular users familiar with the road network.

Several issues of concern related to the utility's geocoding process. Firstly, the dialog from which input table column names representing linear reference elements were specified was ambiguous and confusing. The layout of the dialog appeared to have been modified to provide for different processing options since the original user guide was written. However, the user guide was not updated and the outcomes from several selection options were unclear. Several of the options offered did not perform as expected and several tool buttons performed no function. In addition, the options

offered in relation to the format of linear reference elements were limited (Barney, DK, Clague, SJ & Henry, BA 2008, pers. comm.).

The utility did not provide any form of permanent feedback or status report on the success or otherwise of processes performed on input tables. This made it very difficult to locate and correct invalid data contained in input tables.

Other anecdotal operational problems with Centreline Tools included inexplicable crashes, the display of copious error messages and occasional lapses into endless loops after completing a processing operation. In general terms, the performance and reliability of the utility could be described as inconsistent at best.

3.3 System Analysis

3.3.1 Functional Requirements

The analysis phase of the project involved specifying the functional requirements of the Centreline Tools upgrade. Since this project involved an upgrade to an existing system, the requirements of the upgrade were largely those of the existing version. In fact, retention of the existing functionality and look and feel of the utility were fundamental requirements of the project. To determine the requirements of the new version, an examination of the operations performed by the existing utility was performed.

The existing version of Centreline Tools performs a number of spatial operations designed to provide information related to the state-controlled road network in Queensland. These operations involve translation between linear referenced data in a format based on the Road Reference System (RRS) used in Main Roads, and traditional spatial data based on an earth coordinate reference system. The operations performed by the original version of Centreline Tools are detailed in a user guide (Department of Main Roads 1999).

System operations are selected from a toolbar created when the utility is launched. The toolbar from the original version is shown in Figure 3.2.

Figure 3.2 – Original Centre Line Tools Toolbar

The operation performed by each tool button is described in detail the original utility user guide included in Appendix L. These operations are summarised as follows:

- determining the linear reference description at the location specified by a mouse click on or near the road centreline,
- finding the spatial location on the road network represented by an input linear reference,
- creating spatial objects representing sections of the road network occurring between two points specified by clicking the mouse in the map window,
- creating spatial objects representing sections of the road network occurring between two specified linear references,
- creating point and polyline spatial objects representing the linear references contained in a specified input table,
- generating the linear references represented by point objects contained in a specified input table, and
- displaying and optionally saving spatial objects and information generated during processing.

3.3.2 Additional Requirements

One aspect of the existing system identified for improvement was the information produced during file processing operations. The status of the processing success or otherwise of a particular record is displayed in the message window during processing, but is not permanently saved. Watching for processing anomalies in large files is virtually impossible as is checking each record afterwards. It was therefore proposed to add some form of reporting capability on the status of table processing.

Two options were considered to achieve this outcome:

(a) generating an additional log file, and

(b) adding reporting information to the output file.

A log file is a permanent record but still requires matching records between the log file and the processed output file to derive useful information.

The preferred option was to add a brief comment to the processed output file. The comment would then permanently reside with each processed record and would be instantly available.

3.3.3 Constraints

Centreline Tools is an operational departmental utility. As such, constraints did exist in relation to the functional and operational requirements of the upgrade.

MapInfo Professional is the desktop GIS environment currently in use in Main Roads. As a result, the department has made a significant investment over time in software licence fees, building staff capability and developing customised utilities, of which Centreline Tools is one example. To change would involve re-training staff and redeveloping tools and utilities on another platform, incurring significant costs in time and resources.

Centreline Tools was originally developed in MapBasic to run in the MapInfo environment. It is a widely used utility and performs an important function in the context of Main Roads business.

A number of constraints therefore applied to the Centreline Tools upgrade project, including the following:

- Development in the MapBasic language was mandatory.
- Continued operation with the MapInfo road centreline product *sc_roads* was mandatory.
- The existing general look and feel of interface dialogs and tool buttons was to be maintained for reasons of operational consistency.
- All existing functionality was to be retained.
- Access via the customised Main Roads Menu was to be retained.
- The existing method of distribution as an executable .mbx file was to remain the same.

- Existing library procedures developed for other SAS components could be used.
- For strategic and budgeting purposes, the project was classified within the department as system maintenance.

3.4 System Design

3.4.1 Introduction

In the initial phase of system design, the source code of the existing version of Centreline Tools was examined. The purpose of this was twofold. Firstly, it was necessary to assess the feasibility of redeveloping the utility based on the existing design. Secondly, if this was not the case, to determine if any existing design elements could be used in the new version.

This examination revealed that the logic of the program was convoluted and difficult to follow. The code appeared overly complex and contained very little useful explanatory documentation. A decision was therefore made early in the process that it would be more time-effective to redesign and redevelop the utility than to attempt to re-code the existing design. A redesigned utility would also produce a more reliable and maintainable product into the future.

In spite of the decision to redesign the utility, several significant elements of the existing design were retained in the new version. These design elements are identified in the following sections.

3.4.2 Design Specifications

The design of the Centreline Tools upgrade surrounds the structure and properties of the MapInfo road centreline table '*sc_roads*'. As discussed in Chapter 2, a relationship exists between *sc_roads* and the linear Road Reference System such that values linking the vector objects to the RRS are contained in, or can be derived from, the attributes of the *sc_roads* table.

In order to satisfy the functional requirements of the Centreline Tools upgrade, it was necessary to consider several fundamental processes in the system design. These are:

- to determine the spatial location of the point on the road network at which a given linear reference occurs,
- to determine the linear reference at the point on the road network represented by a given spatial location, and
- to expand these single location based processes for use on sections of the road network.

The remaining functional requirements involving interactive program input and administrative tasks related to generated data had to be developed for the upgrade, but did not require any significant redesign.

3.4.3 Determining the Spatial Location of a Given Linear Reference

The initial phase of the design was concerned with identifying the specific road centreline segment in the *sc_roads* table in which a given linear reference occurs.

In order to describe a unique location on the road network, a linear reference must contain, at a minimum, the identity of the road and a measurement along the road from some point of reference, usually the starting point of the road. This measurement is known as ‘chainage’ or ‘through distance’. Where the form of the road is more complex, a carriageway code is also necessary to describe a unique location.

The attribute values in the *sc_roads* table therefore provide the key to the identification of the required segment, as follows:

- The identity of the road is contained in the STREET attribute.
- The carriageway code is contained in the CARRWAY attribute.
- The chainage at the start of the segment is contained in REFOFFSET.
- The chainage at the end of the segment is equal to the start chainage plus the driven length of the segment (*REFOFFSET* + *REFLEN*).

In combination, these attribute values provide the identity of the specific centreline segment within which the given linear reference occurs.

Once the required segment has been identified, the next step in the design strategy involves identifying the 'link' in the segment object within which the chainage component of the given linear reference occurs. A link is the straight section of the vector object between two adjacent nodes (see Chapter 2 for a description of MapInfo vector structures).

The methodology involves traversing the segment object programmatically, one link at a time from its starting point. Since the chainage at the segment start point is known, an accumulating chainage can be calculated by converting the length of each link to chainage using the SCALEFACTOR attribute (see Chapter 2 for a description of the disparity between driven length and object length). When the accumulating chainage value exceeds the given chainage, the given chainage occurs somewhere in the current link. The given chainage therefore occurs in the segment at a distance along the link equal to the difference between the given chainage and the accumulated chainage at the start of the current link. This value is the remaining chainage.

The location of the given linear reference can be represented by a circular vector object centred at the start node of the current link, with a radius equal to the remaining chainage. The given linear reference will occur at the intersection point between the circular object and the segment object.

The intersection point is then be used to create a spatial point object to represent the location of a linear reference.

One significant issue complicates the design strategy described above. As described in Chapter 2, chainage increases in the direction in which roads are gazetted. However, some carriageway types, and hence the vector objects representing them, have a direction that is opposite to, or against road gazettal direction. In these cases, vector object nodes increase in number against gazettal. Therefore, to traverse these segment objects in the direction of gazettal requires commencing at the last node, and addressing the nodes in descending order towards node number one. Object direction and node numbering were therefore important considerations in the development process.

3.4.4 Determining the Linear Reference of a Given Spatial Location

In Centreline Tools, a given spatial location is represented by a point object. However, the point object may or may not be coincident with a road centreline segment, making identification of the required segment problematic.

The design solution used to identify the segment object proximal to the given point was retained from the existing version. An expanding circular area object, centred at the spatial coordinates of the given point object was utilised. As the area object expands, it will intersect with a road segment object, if the given point at its centre is in the proximity of the road network.

Once the required segment object has been identified, the location in the segment that is closest to the given point must be found. At this stage the design strategy for the upgrade diverged significantly from the original version. The new design involves the construction of two circular geometric objects of equal radii, centred at the intersection points of the expanding area object and the centreline segment object. A line object is then created between the intersection points of the two circular objects. This line must intersect the segment object.

It then remains to sequentially traverse the segment object links to identify the link that is intersected by the constructed line object. Once this link is identified a new node can be placed in the segment object at the intersection point. The length of the segment object up to the new node is then measured and converted to chainage using the SCALEFACTOR correction value.

The centreline segment attribute values of road identifier (STREET) and carriageway type (CARRWAY) and then combined with the calculated chainage to provide the linear reference at the given point.

A point object can then be created with the coordinates of the new node to represent the precise spatial location of the determined linear reference.

As in the previous section, segment object direction in relation to road gazettal direction, and consequential sequencing of nodes significantly complicated the development of this design concept.

3.4.5 Constructing Road Sections

The design strategies described above are suitable for translating between individual spatial locations and linear references. However, the specifications of Centreline Tools require the implementation of true dynamic segmentation. This involves creating spatial objects to represent sections of road as well as single locations.

Constructing road sections commences with the identification of all the centreline segment objects that occur between the defined linear extents of the section. The required segments can be identified by the attribute values contained in the *sc_roads* table, as in the previous section.

To support the construction of spatial objects representing sections of road, the concept of a ‘portion’ was introduced. For the purposes of this project, a portion is defined as that part of an individual centreline segment object that occurs within a defined road section. A portion may therefore consist of an entire segment object, or the remains of one truncated at the chainage defining the start and or end of the road section.

The design strategies used to construct portion objects based on linear references representing the extents of a road section are similar to those described previously and involve the programmatic traversal of segment objects.

An object representing a road section is constructed by combining all the segment object portions that occur between the start and end of the defined section.

3.5 System Development

The system development phase of the Centreline Tools upgrade project is presented in detail in Chapter 4, and describes the coding methods used to implement the design strategies introduced in the previous section.

3.6 System Implementation

3.6.1 Installation

In the implementation phase of the project, a completed development version of the Centreline Tools upgrade was distributed for installation and operational testing. The program executable file CT_Main.MBX was attached to an e-mail, which contained release notes and installation instructions.

Installation involved adding a new item for the Centreline Tools upgrade to the MainRoads MapInfo Menu using the Menu Customisation Dialog. This dialog is accessed from the MainRoads menu as shown in Figure 3.3. The name of the application, the directory path to the executable file and help string to appear in the MapInfo Status Bar were entered into the customisation dialog. The entry for Centreline Tools in the customisation dialog is shown in Figure 3.4. This procedure was necessary for the initial release only.

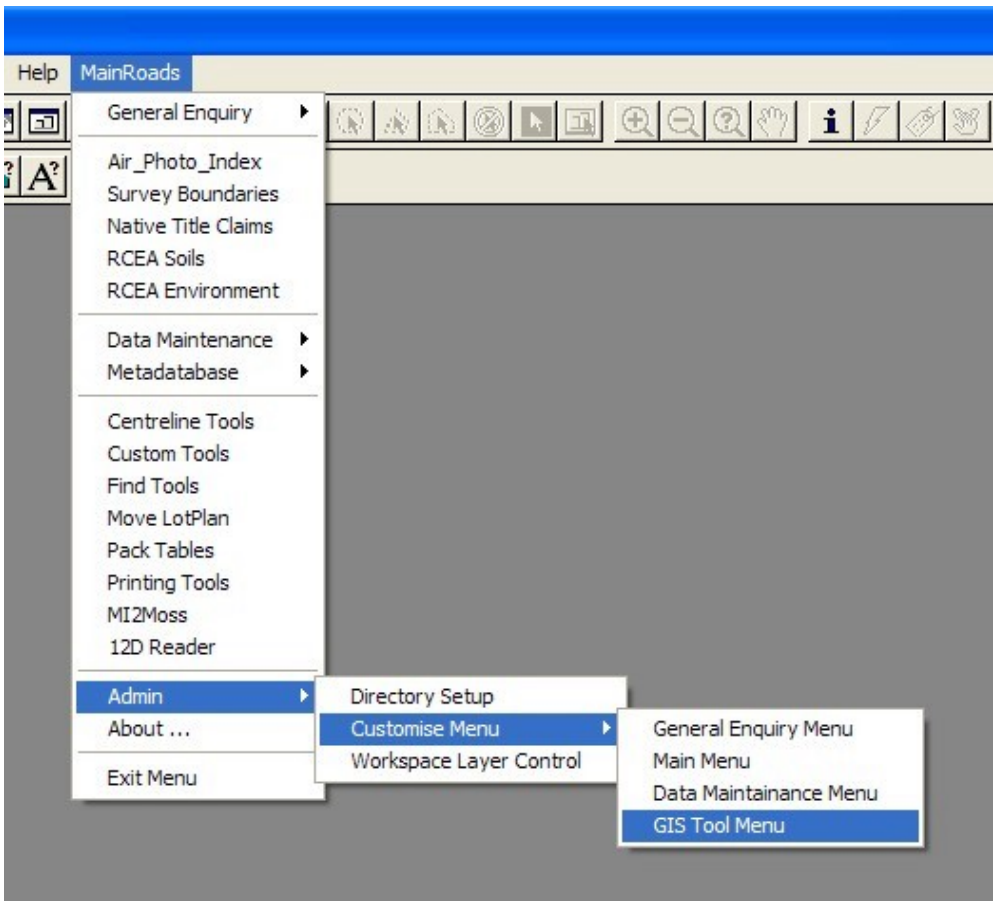


Figure 3.3 – Main Roads MapInfo Menu

The installation process is completed by placing the executable file CT_Main.MBX in the network directory reserved for SAS utilities. This operation had to be repeated for each subsequent release.

The new release of Centreline Tools could then be launched by a single mouse click from the MainRoads Menu. Figure 3.5 shows how Centreline Tools appears in the MainRoads Menu, ready for selection.

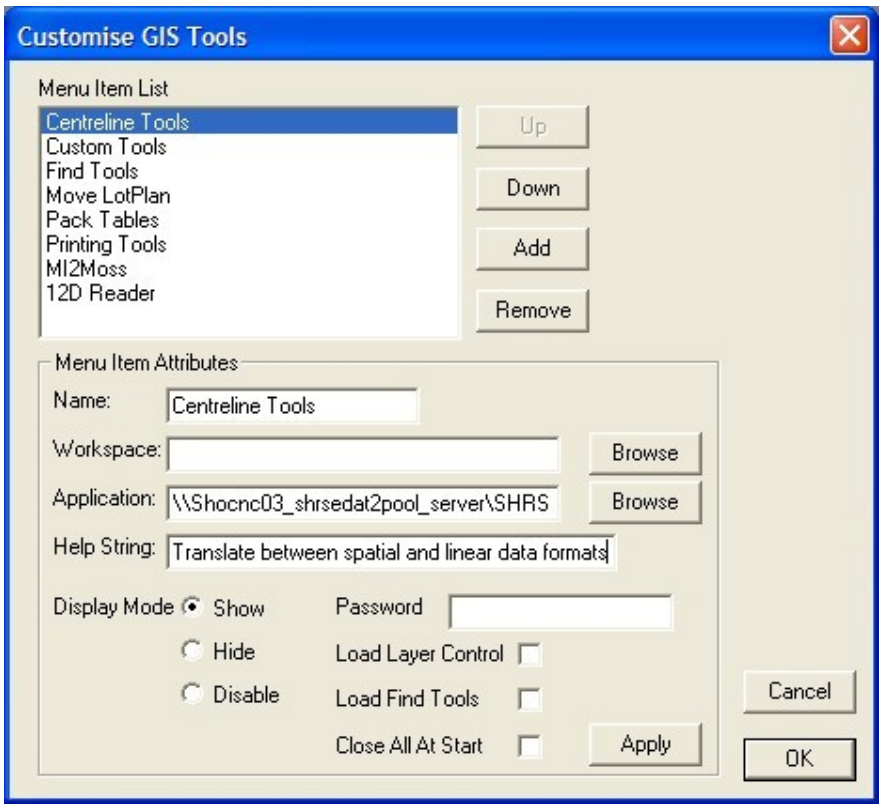


Figure 3.4 – Main Roads Menu Customisation Dialog

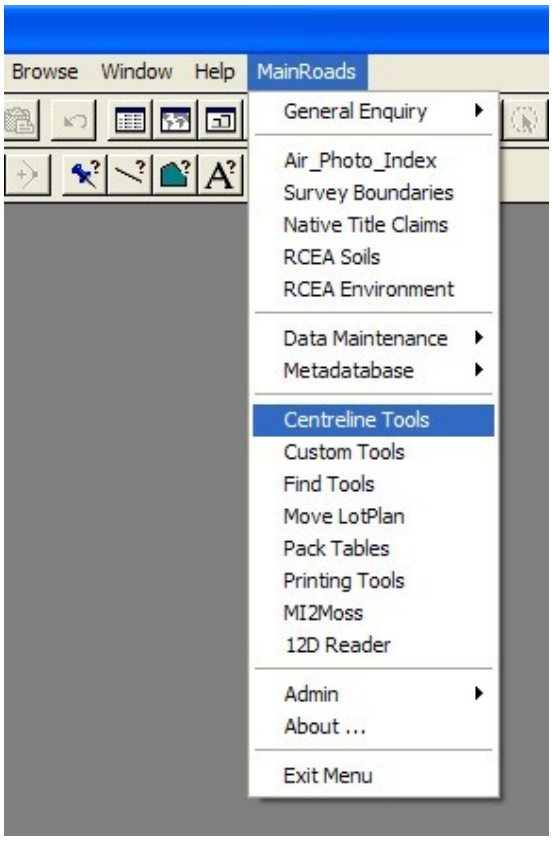


Figure 3.5 – Menu Selection of Centreline Tools

3.6.2 Training

Because the Centreline Tools upgrade was necessarily very similar in function and operation to the old version, very little training was required.

However, some changes in the operation and behaviour of the upgrade did require explanation. Users were made aware of these changes in release notes that were sent with the executable .mbx file, initially to the test group, and finally to the regional GIS coordinators when the production release was distributed.

Additional information sessions are proposed in the future. A new version of the Centreline Tools user guide is also proposed.

3.7 System Maintenance

As well as testing during the development phase of the project, individuals within the department were specifically requested to test development versions of the Centreline Tools upgrade in an operational environment in regional offices.

The test group was contacted regularly to assess progress, provide support and gather feedback on the operation and reliability of the development versions of the upgrade. Feedback was evaluated and subsequently incorporated into the development phase of the project.

This process resulted in the identification and correction of several programming errors and modifications to the requirements of the utility.

CHAPTER 4
SYSTEM DEVELOPMENT

4.1 Introduction

In the system development phase of the project, several identifiable although not necessarily discrete activities were undertaken. These activities consisted of:

- assessing the source code of the existing version to identify elements suitable for reuse in the upgraded version,
- writing new MapBasic functions and sub-procedures to implement the design concepts introduced in Chapter 3, and
- combining the new material with modified existing code to produce an upgraded version of Centreline Tools.

In reality, these activities merged into one process. The development phase of the Centreline Tools upgrade project is presented in the following sections.

4.2 Program Structure

4.2.1 Overview

Centreline Tools is a structured collection of sub-procedures and functions written in the MapBasic language, which are loosely grouped into modules on the basis of the operation they perform. The source code modules are 'Included' into the main module, which is compiled directly to an executable file called CT_Main.MBX. All the modules

used are specific to Centreline Tools, with the exception of two that contain library files used to support other SAS utilities.

Figure 4.1 is an extract from the module CT_Main.MB and shows all of the program modules used in the Centreline Tools upgrade.

```
'-------------------------------------------------------------------------------
' Modules used in the Centreline Tools utility.
' See utility definition file for definitions and declarations.
'-------------------------------------------------------------------------------
INCLUDE "MapBasic.def"                  '- standard MapBasic definitions
INCLUDE "Icons.def"                     '- standard MapBasic definitions
INCLUDE "Menu.def"                      '- standard MapBasic definitions
INCLUDE "CT_Def.DEF"                    '- centreline utility definition file
INCLUDE "..\..\_Common\Lib_ini.MB"      '- SAS ini procedure library
INCLUDE "..\..\_Common\Lib_funct2.MB"   '- SAS procedure library
INCLUDE "CT_CoordsToRef.MB"             '- translates input points to chainages
INCLUDE "CT_Debug.MB"                   '- utility debugging routines
INCLUDE "CT_Dialogs.MB"                 '- dialogs for table/process selection
INCLUDE "CT_DynSeg.MB"                  '- manages dynamic segmentation
INCLUDE "CT_FindNodes.MB"               '- node finding and adding procedures
INCLUDE "CT_RefToCoords.MB"             '- translates input chainages to points
INCLUDE "CT_SectionByCoords.MB"         '- extracts a section by coordinates
INCLUDE "CT_SectionByRefs.MB"           '- extracts a section by linear references
INCLUDE "CT_Select.MB"                  '- contains queries from sc_roads
'-------------------------------------------------------------------------------
```

Figure 4.1 – Code Modules Used in Centreline Tools

4.2.2 The Definition File

A new program structure was introduced for the Centreline Tools upgrade. In the existing program, procedures were generally declared in the module in which they were coded. This arrangement contributed to difficulties in following the program's logic and caused potential compilation problems because of the order in which procedures were declared, and modules included in the main module. In addition, the MapBasic definition files MapBasic.def, Icons.def and Menu.def were included in each module.

In the upgrade, all procedures were declared in the program definition file, CT_Def.DEF with the global variable declarations and constant definitions. This strategy centralised information for easier reference during development and maintenance. It also avoided the refer-before-declare issues caused by procedure declarations in individual modules. As the modules relate only to this utility, a requirement for each module to compile independently was considered unnecessary.

New program constants were introduced for version identifier, release date, and road centreline table name. The use of constant values simplifies maintenance since a change to any of these values requires one change only in the definition file.

The complete Centreline Tools definition file, CT_Def.DEF is included in Appendix B.

4.2.3 The Main Module

When Centreline Tools is launched, program execution commences with the sub-procedure called Main.

Calls to sub-procedures ReadInitSetup and WriteLogEntry are used in the majority of SAS utilities, and were included in the Centreline Tools upgrade. They respectively read MainRoads Menu System directories from an initialisation file into global variables for reference purposes, and write an entry to an activity log file.

Because Centreline Tools is dependent on a specific road centreline table, the function OpenClTab(), renamed and slightly modified from the existing version, was used to verify that the table *sc_roads* is open, or request it to be opened if it is not. The sub-procedure DisplayClTab displays *sc_roads* in a new MapInfo map window, if one is not open.

The most significant task to be performed by the Main sub-procedure was to construct the Centreline Tools Toolbar, from which all utility operations are launched or selected. MapBasic statements Create ButtonPad, ToolButton and PushButton were used to construct the toolbar.

Structure charts for procedures developed for the Centreline Tools upgrade, including the Main sub-procedure, are included in Appendix C.

4.3 Selecting and Filtering Road Centreline Segments

4.3.1 Selecting Segments for a Linear Reference

As described in Chapter 3, identifying specific road centreline segment objects contained in the *sc_roads* table was fundamental to performing spatial to linear and linear to spatial translations.

Road centreline segments were selected using 'Select' statements, which are essentially a MapBasic implementation of SQL queries. Select statements produce a temporary MapInfo table containing the records that match the expression in the 'Where' clause of the statement. Where possible, select statements on *sc_roads* were grouped into the module CT_Select.MB.

The function FindReferenceSegments() was developed to select the road segments in which a single specified linear reference occurs. That is, segments with the specified road identifier, and having a start chainage less than or equal to the specified chainage, and an end chainage greater than or equal to the specified chainage. The use of a greater than operator (>) could not be used here because end segments (ie segments with no adjoining segments) would not be selected. The following extract from the function FindReferenceSegments() shows the MapBasic select statement:

```
Select * From CT_SC_ROADS
Where STREET = sRoadId AND
      CARRWAY = sCarriageway AND
      REFOFFSET <= fChainage AND
      REFOFFSET + REFLEN >= fChainage
Into tblReferenceSegs Noselect
Order By CARRWAY, REFOFFSET
```

This statement creates the temporary MapInfo table tblReferenceSegs which contains all the road segment records in which the chainage value in the variable fChainage occurs, including segments for multiple carriageways, if they exist.

Variations on the value of the carriageway code variable sCarriageway in the function enabled the selection of segments for various combinations of carriageway types.

Selected segments were ordered by carriageway type (CARRWAY) and segment start chainage (REFOFFSET) to facilitate processing elsewhere in the program.

The function then assigns the selected segments to a SegmentType array variable (see below). Unwanted segments are filtered out during this process. Segment filtering is discussed in Section 4.3.3.

The data type 'SegmentType' was introduced in the upgrade to contain a road centreline segment object and the set of attributes used in format translations. The use of a defined date type had several benefits. Firstly, it made procedures involving road segment manipulations more independent. Selected segments can be stored and passed between procedures as array variables of type SegmentType. This method avoids continual reference to selections made in other procedures, which was the method employed in the existing version. Continual reference to selection tables disrupts program continuity and is similar to the overuse of global variables. Neither strategy was favoured in the development of the upgrade.

Secondly, the defined data type can potentially be used for segment selections from any road centreline table containing Road Reference System based attributes. In the event of the introduction of another road centreline product, only the select statement would need to be modified, significantly reducing future maintenance requirements. Figure 4.2 shows the type definition statement used for the SegmentType data type.

```
TYPE SegmentType
   SegObj      As Object          '- segment object
   SegRoad     As String          '- segment road identifier
   SegCway     As String          '- segment carriageway code
   SegStart    As Float           '- segment start chainage (in m)
   SegLength   As Float           '- segment driven length (in m)
   SegScale    As Float           '- segment scalefactor
END TYPE
```

Figure 4.2 – Definition Statement for the SegmentType Data Type

Appendix D contains the entire function FindReferenceSegments() and illustrates the use of select statements. The function also demonstrates the use of a variable to control the selection of segments representing various combinations of carriageway types.

4.3.2 Selecting Segments for a Spatial Location

The function FindPointSegments() was developed to select road segment objects in proximity to a specified spatial location. An expanding circular area object, centred at the coordinates of the specified location, was used to locate proximal segment objects. The area object is expanded in preset increments until it intersects a segment object, or a maximum search radius value represented by a system constant is reached. A select statement with a spatial condition in the Where clause was used as the following extract from the function FindPointSegments() shows:

```
Select * From CT_SC_ROADS
Where oSearchArea Contains Part Obj
Into tblPointSegs Noselect
Order By STREET, CARRWAY, REFOFFSET
```

The select statement was placed in a loop based on the incrementing value of the search area radius. If a segment is selected, the loop is terminated. After segments are selected, the search area is expanded by one additional increment value to ensure that the intersection is not a tangential (single point) intersection. Examples of search area intersections with road segments are illustrated in Figure 4.3.

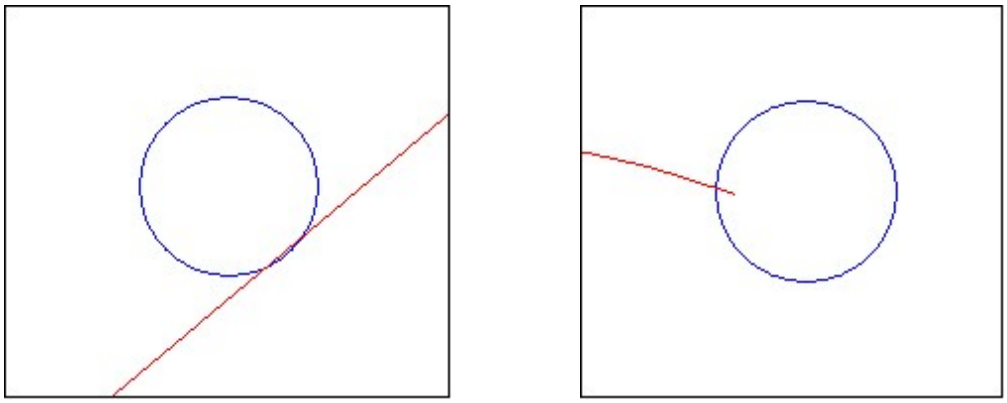


Figure 4.3 – Search Area Intersections with Road Segments

In this selection process, the road identifier is not known as it is in the selection for a linear reference, so the result table tblPointSegs might contain segments from multiple roads. The segments selected are therefore ordered by road identifier (STREET), carriageway type (CARRWAY), and segment start chainage (REFOFFSET) to assist filtering and processing for multiple roads if necessary.

The selected segments are then filtered and assigned to a SegmentType array variable as in the previous section.

4.3.3 Filtering Selected Segments

A linear reference consisting of a road identifier, carriageway code and chainage value represents a unique location on the road network (excluding lane structures). However, in some circumstances, more that one road segment satisfies the condition clauses used in the statements described in pervious sections. It was therefore necessary to filter out (remove) redundant segments prior to processing.

This problem arises when a specified chainage occurs at a Reference Point (RP). When an RP occurs on a given carriageway, the end point of one segment, and at the start point of the adjoining segment are coincidental with the RP, and consequently have the same chainage value. The statement therefore selects the segment with a start chainage (REFOFFSET) equal to the specified chainage as well as the segment with an end chainage (REFOFFSET + REFLEN) equal to the specified chainage. A RP on a single carriageway road is shown in Figure 4.4. In these cases the select statement will select the segment on each side of the RP.

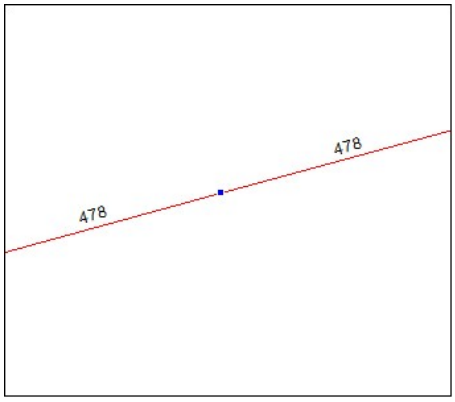


Figure 4.4 – Reference Point on a Single Carriageway

Redundant segments are not selected at every RP. Figures 4.5 and 4.6 demonstrate how RPs can occur in different circumstances in relation to the road network. In Figure 4.5, filtering is required for Road 426. In Figure 4.6 however, no filtering would be required.

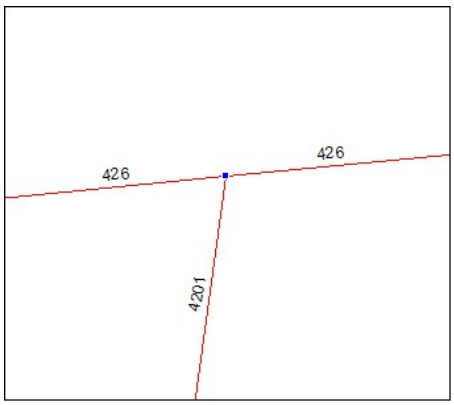


Figure 4.5 – Reference Point at an Intersection

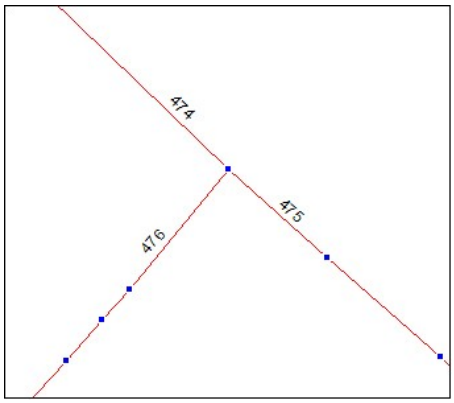


Figure 4.6 – Reference Point on Multiple Roads

The use of a less than operator (<) in selections may have avoided the need for filtering. However, the less than or equal to operator (<=) was necessary to ensure that objects representing the end segments of roads were included in the selection (ie in situations where there is no contiguous segment, as shown in Figure 4.7).

Segment filtering was implemented by assigning the first segment record in the result table to the SegmentType array variable, then testing each subsequent record against the last for road/carriageway equality, prior to assignment. Duplicates are ignored. Appendix D illustrates how segment filtering is implemented in the function FindReferenceSegments().

Applying this filtering strategy ensures that the SegmentType array variable only ever contains segments with unique road and carriageway combinations.

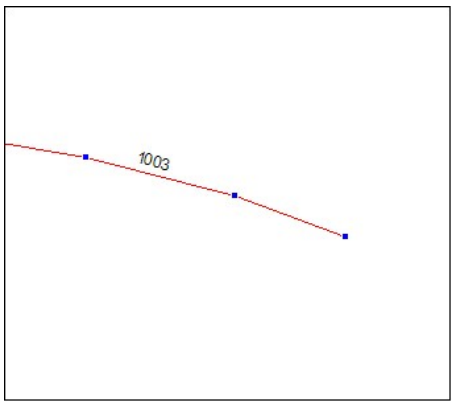


Figure 4.7 – End of Road Segment

4.3.4 Selecting Segments for Road Sections

Selections for road sections are more complex because sections may comprise multiple road segments. The selection of road segment objects that make up a specified section of road was coded into the function FindSectionSegments(), as shown in the following extract:

```
Select * From CT_SC_ROADS
Where STREET = sRoadId AND
      CARRWAY = sCarriageway AND
      REFOFFSET + REFLEN > fSectionStart AND
      REFOFFSET < fSectionEnd   '- <= includes the next segment
Into tblSectionSegs Noselect
Order By CARRWAY, REFOFFSET
```

In this statement, the less than or equal to operator (<=) is not used because it includes the next contiguous segment in the result table. The next contiguous segment is not wanted in this case.

This statement is similar to the selection for a single linear reference. However, it contains two chainage variables, fSectionStart representing the beginning of the section, and fSectionEnd representing the end of the section. All the segments occurring between the linear references defining the section are therefore selected.

The selected segments are ordered by carriageway (CARRWAY) and increasing start chainage (REFOFFSET).

Segments representing various combinations of carriageway types were selected by varying the value of the carriageway code variable sCarriageway in the function.

The selected segments are assigned to a SegmentType array variable but filtering was not necessary because adjacent segments may be required to construct a road section.

4.3.5 Other Selections

Select statements were also used to validate input data and extract additional information from the road centreline table *sc_roads*.

The function RoadExists() was used to verify that a named road identifier is valid, as the following extract shows:

```
Select STREET From CT_SC_ROADS
Where STREET = sRoadId
Into tblRoadExists Noselect
```

If the result table tblRoadExists is empty following the selection, the road represented by the String variable sRoadId is not valid.

A selection statement was used in the function RoadExtents() to acquire the minimum and maximum chainages of a named road identifier, as shown in the following extract:

```
Select Min(REFOFFSET)           "RoadStart",
       Max(REFOFFSET + REFLEN) "RoadEnd"
From CT_SC_ROADS
Where STREET = sRoadId
Into tblRoadExtents Noselect
```

The aggregate functions Min() and Max() were used to find the minimum and maximum start and end chainages respectively of the entire road represented by the String variable sRoadId. The aggregate function return values are placed into the columns RoadStart and RoadEnd of the result table tblRoadExtents.

4.4 Processing Segment Objects

4.4.1 Segment Direction

The direction of road segment objects was a critical consideration in the development of the Centreline Tools upgrade. Segment direction is a function of gazettal direction and carriageway code. To facilitate the use of object direction in the code, program constants representing the direction of specific carriageway types, and the direction of segment objects in relation to gazettal direction, were retained from the existing version. Figure 4.8 is an extract from the definition file CT_Def.DEF showing the constant definitions for carriageway types and relevant object direction.

```
DEFINE CT_CWAY_BOTH         "1"            '- cway code in both directions
DEFINE CT_CWAY_WITH_GAZ     "24ABCDEKLMXZ" '- cway codes in gazettal direction
DEFINE CT_CWAY_AGAINST_GAZ  "35NOPQRSTUY"  '- cway codes against gazettal
DEFINE CT_DIR_WITH_GAZ       1             '- line direction with gazettal
DEFINE CT_DIR_AGAINST_GAZ    2             '- line direction against gazettal
DEFINE CT_DIR_UNKNOWN        3             '- line direction unspecified
```

Figure 4.8 – Constant Definitions Representing Direction

The function SegmentDirection() was retained from the existing version. This function, compares a given carriageway code with the String constants shown above, and returns a small integer value representing the direction of the segment object based on its carriageway type.

4.4.2 Finding the Spatial Location of a Chainage

The function FindNodeAtChainage() finds the number of the node in a selected segment object at which a given chainage occurs. The required node may be a new node created by the function, or an existing node if one occurs at the chainage.

The given chainage can occur at the start point or end point of the segment, or some point in between. If the given chainage is equal to the attribute value representing the chainage at the start of the segment (REFOFFSET), the chainage occurs at the first node of the object (node number one), if the object direction is with gazettal. If the direction of the object is against gazettal, the chainage occurs at the last node (maximum node

number) of the object. The number of nodes in the object is retrieved using the MapBasic function ObjectInfo(SegmentObject, OBJ_INFO_NPNTS), where SegmentObject represents the segment object and OBJ_INFO_NPNTS is a MapBasic constant.

Similarly, if the given chainage is equal to the chainage value at the end of the segment (*REFOFFSET* + *REFLEN*), the chainage occurs at the last or first object node depending on object direction.

If the given chainage occurs elsewhere within the segment, the function loops sequentially through the segment object link by link (node-node; node-node), searching for the link in which the chainage occurs. If the segment object direction is with gazettal, the loop begins at node number one and progresses in ascending node order. Otherwise it begins at the maximum node and descends towards node number one. The MapBasic function ExtractNodes() was used in the function to create an object to represent a link from two adjacent node numbers. Node number pairs are incremented or decremented in the loop to produce a sequential series of link objects.

At the beginning of each loop, the length of the current link is extracted by the MapBasic function ObjectLen(), and subtotalled to represent the length of the segment object traversed up to the current iteration of the loop. The object length traversed is converted to chainage traversed, by dividing the subtotal by the SCALEFACTOR attribute value. When the chainage traversed exceeds the given chainage, the given chainage must occur at some point within the current link of the segment object. This point occurs beyond the start of the current link, at a chainage equal to the difference between the chainage at the start of the link, and the given chainage.

To determine this point in the link object, the chainage excess beyond the segment start point is converted back to distance, this time by multiplying the excess chainage by the SCALEFACTOR attribute value. The MapBasic function CreateCircle() was used to create a circular object, centred at the first node of the current link object, with a radius equal to the excess distance. This circle object must intersect with the link object because the given chainage falls within the link. An example circle/link intersection is demonstrated in Figure 4.9.

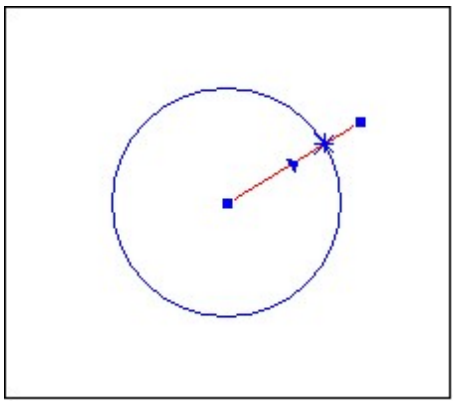


Figure 4.9 – Circle Intersection with a Link Object

The MapBasic function OverlayNodes() was then used to create another object similar to the link object, but with an additional node at the intersect point with the circle object. A new node is then be added to the segment object variable using the coordinates of the node at the intersect point. This new node in the segment object variable is the point at which the given chainage occurs. The number of the new node is the one after, or the one before the segment node used for the start of the current link object, depending on the direction of the segment object.

The loop is terminated when the required link is found and the function FindNodeAtChainage() returns the number of the new node. The spatial coordinates at the specified linear reference are therefore those of the new node in the segment.

4.4.3 Finding the Chainage at a Spatial Location

The function FindNodeAtPoint() returns the node number of the node within the selected segment object that is as close as possible to the spatial location represented by a given point object.

The segment object closest to the point object is selected using the expanding area spatial selection process described in Section 4.3.2. Following the selection of an intersecting segment, the circular object known to intersect the segment is used to locate the spatial location in the segment closest to the given point object.

The algorithm used to implement this process was based on simple geometric construction, as would be done manually using intersecting arcs to bisect a line, and is described in the following five steps.

Step 1.

The location of the search area is tested in relation to the segment object. Point objects are created at the coordinates of the start and end nodes of the segment object. A spatial select statement determines if one of these points falls within the search area. If so, the number of the node concerned is assigned to the function for return. In this way the given point object is effectively 'snapped' to an end point of the segment, as shown in Figure 4.10. Segment object direction is not a concern for this step.

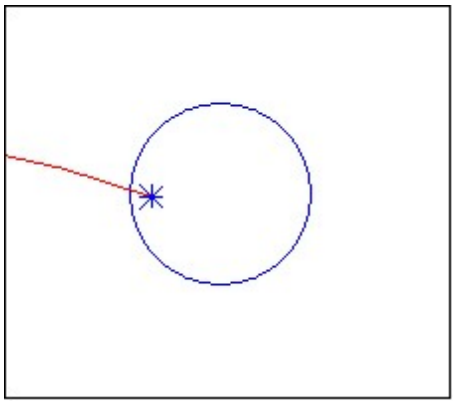


Figure 4.10 – Segment Object End Node Assignment

Step 2.

If neither end of the segment object falls within the search area, an object representing the portion of the segment object that is intersected, is created using the MapBasic function Overlap(). The area/segment object intersection and resulting 'overlap' object are illustrated in Figures 4.11 and 4.12.

Step 3.

Two circular objects, centred at the coordinates of the end points of the 'overlap' object, and with radii equal to its length, are created. These circular objects, shown in Figure 4.13, must intersect because their radii are equal.

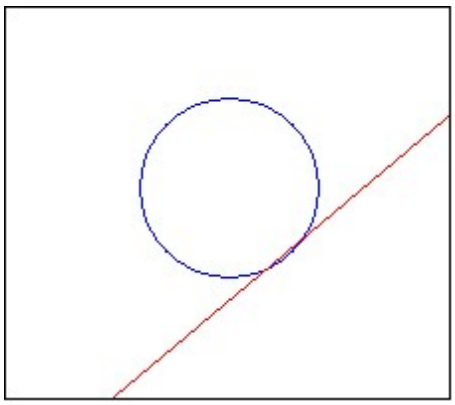


Figure 4.11 – Search Area Intersection with a Segment Object

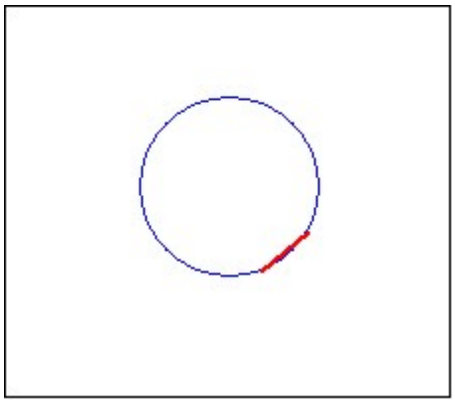


Figure 4.12 – Overlap Object from Intersection

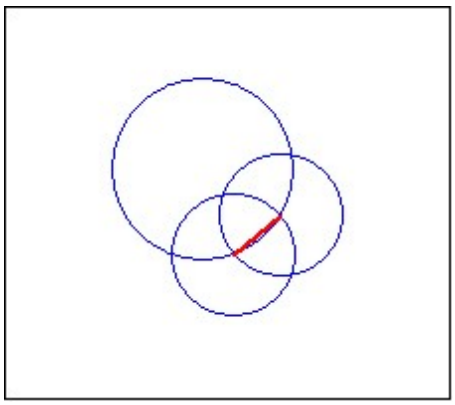


Figure 4.13 – Intersecting Circle Objects

Step 4.

The MapBasic function IntersectNodes() was used to create a polyline object joining the intersection points of the two circles created in the previous step. This polyline object, shown in Figure 4.14, must intersect the segment object, and does so at the point on the

segment that is closest to the centre of the search area, which in turn represents the given spatial location.

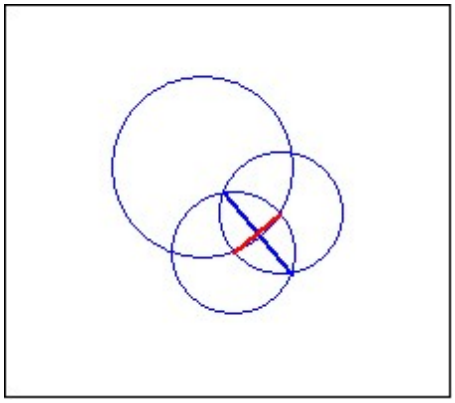


Figure 4.14 – Polyline Joining Circle Intersect Points

Step 5.

The intersection point between the polyline object and the segment object is located by looping sequentially through the links of the segment object, starting at node number one, searching for the link object that is intersected by the polyline object (segment direction is not relevant here). The intersection point in the segment object is shown in Figure 4.15. The method used to loop through segment link objects was described in the previous section.

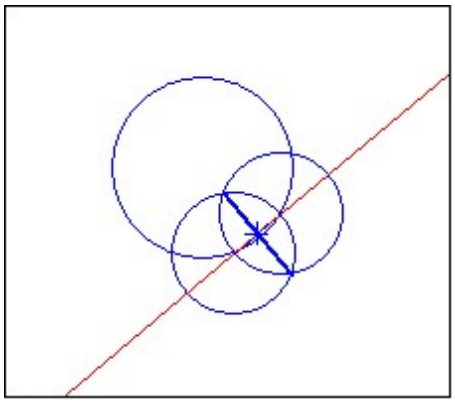


Figure 4.15 – Intersect Point in the Segment Object

The MapBasic function OverlayNodes() was then used to construct another polyline object with a node at the intersection point between the original polyline object and the link object. The spatial coordinates of this new node are stored and compared with that of each end of the link object. If a match occurs, the node number of the segment object

node providing the link is assigned to the function. Otherwise, the intersection point occurs elsewhere in the link and a new node is added to the segment object using the stored coordinates, and the new node number is assigned to the function.

The node number returned by FindNodeAtPoint() therefore represents the location within the segment object that is as close as possible to the actual location of the original point object.

The Function ChainageAtNode() determines the chainage at a given node number in a segment object. The given node can be the first node of the segment object, the last node, or any node in between. If the given node is node number one, the required chainage will be the start chainage of the segment (REFOFFSET), provided the segment object direction is with gazettal. If the direction of the object is against gazettal, the chainage at node number one will be the chainage at the end of the segment (*REFOFFSET* + *REFLEN*). This scenario is reversed if the given node is the last node of the segment.

Where the given node occurs elsewhere in the segment, the MapBasic function ExtractNodes() was used to extract the portion of the segment for which the chainage calculation is relevant. If the segment object is in gazettal direction, the portion extracted is between node number one and the given node. Otherwise, the portion is between the given node and the last node.

The length of the object extracted from the segment is measured by the MapBasic function ObjectLen(), and then converted to chainage using the SCALEFACTOR attribute, as described previously.

The chainage value at the given node is then assigned to the function and returned to the calling procedure.

4.4.4 Constructing Sections

Dynamic segmentation involves the construction of spatial vector objects representing defined sections of a road.

An individual road segment object can relate to a defined section of road in a number of ways. That is, a segment object can:

- begin before the start of the section and finish after the end of the section.
- begin before the start of the section and finish at or before the end of the section.
- begin at or after the start of the section and finish after the end of the section.
- begin at or after the start of the section and finish at or before the end of the section.

The construction of a defined road section therefore involves identifying the part of each individual road segment object that occurs within the confines of the section. For the purposes of this project, this part of a segment object is called a 'portion', as discussed in Chapter 3. The function ConstructPortion() was written to construct a portion vector object from a given segment object.

The function ConstructPortion() tests for each possibility of the location of the segment in relation to the section. It determines the node numbers at the chainages defining the road section using the function FindNodeAtChainage(), described in Section 4.4.2, and the SegmentDirection() function, also described previously. Adding nodes in this function is more complicated than previously described. If a segment object is cut at both ends, a second node must be added to the segment object, necessitating a double shift in node numbers. Node numbering is further complicated by segment object direction.

The portion object is created using the MapBasic function ExtractNodes() with the node numbers representing the section start and end points. The constructed portion is then assigned to the function ConstructPortion().

A road section is constructed by applying the function ConstructPortion() to each of the segment objects selected for the defined road section, and combining the resulting

portions into a single object. Segment object selections for road sections were discussed in Section 4.3.4.

The function ConstructPortion() is used by the dynamic segmentation sub-procedure ProcessReferenceTable and by the sub-procedure ExtractSection which constructs defined sections.

4.5 Processing Tables

4.5.1 Introduction

The system requirements of the Centreline Tools upgrade included the capability to perform segmentation operations on entire MapInfo tables. Two types of tables were involved.

1 Tables containing linear references

Tables of this type normally contain road asset data extracted from ARMIS, or data collected along the road corridor using a vehicle speedometer to obtain chainage values. Processing these tables involves creating vector objects to represent the linear references.

2 Tables containing point objects

Such tables usually contain data collected along the road using a GPS device. In these tables, the linear references representing the coordinate-based locations are determined.

The following sections describe the development of table processing in the Centreline Tools upgrade.

4.5.2 Tables Containing Linear References

The sub-procedure ProcessReferenceTable was developed to manage the processing of tables containing linear references.

Dialogs were used as the means to select an input table and specify a name and path for the processed output table. A MapBasic 'SaveAs' dialog was used to copy the input table to a named output table. The copied table is then processed leaving the original input table unmodified.

The table to be processed must be mappable in order to store the objects created. The following MapBasic statement extracted from the ProcessReferenceTable procedure was used to make the table mappable and sets the table projection to longitude/latitude, GDA94:

```
Create Map For sOutputTable CoordSys Earth Projection 1, 116
```

If the table is already mappable, this statement destroys any objects it contains.

An additional column is added to the table for a processing status report. The following MapBasic statement was used to add a character field of 50 characters to the table:

```
Alter Table sOutputTable (Add Status Char(50))
```

Once the output table has been prepared, information about the table must be gathered for processing. Firstly, the format of the linear reference in the table must be specified. Three linear reference formats were defined as:

- a specific location defined by a road identifier and a single chainage value,
- a section of road defined by a road identifier, and section start and end chainages, and
- a section of road defined by a road identifier, a section start chainage and the length of the section.

Secondly, the column of the table containing each of these linear reference elements must be identified.

A dialog was designed for the interactive entry of the relevant table information. A mutually exclusive RadioGroup Control was used for the selection of the linear reference format.

A separate section of the dialog was set aside for each linear reference format, to be activated when the format is selected. Each section of the dialog has ListBox Controls for the table columns containing the values needed for the specific format. The ListBox Controls are populated with the names of the table columns for selection. Column names are filtered so that only columns containing the required data type are displayed.

RadioGroup Controls were included for the specification of units of measurement for distance values.

The Do While NOT EOT() construct was used to loop through the records in the table. The function EOT() (End of Table) returns TRUE when the end of the table is encountered and the loop terminates.

The table is processed on the basis of the linear reference format and measurement units specified in the entry dialog. The road identifier and chainage values are assigned to local variables from the specified columns. Chainage values are converted from kilometres to metres if necessary.

The road identifier and chainages are tested for validity. A brief message describing any anomalies is assigned to the String variable sReport.

If a chainage is beyond the extent of a road, the value is reset to the maximum chainage of the road and a report is assigned to the sReport variable.

Likewise, section start and end chainages are tested to verify that they are in gazettal direction for processing. The values are swapped if necessary, and a message assigned to the sReport variable.

The procedure also tests for the selection of a carriageway code column. If no carriageway column is specified, segmentation is performed on through carriageways only (ie carriageways 1, 2 and 3). If a carriageway column is specified, the value contained in the selected column is used.

Irrespective of the data format selected, the program algorithm uses a start and end value in each case to construct a section object. If the data format is a single chainage,

the section has a length of zero. Similarly, a section with the same start and end chainages also has a length of zero.

If the section has a length of zero, a point object is created. Otherwise, a section is created using the ConstructPortion() function described previously. In either case the object created is assigned to the object variable oSectionObj.

If no anomalies are encountered during processing, the message 'OK' is assigned to the sReport variable.

The following MapBasic statements, extracted from ProcessReferenceTable, were used to update the row of the table currently being processed with the content of the variables oSectionObj and sReport:

```
Update tblReferences Set Obj = oSectionObj
Where RowID = tblReferences.RowID

Update tblReferences Set Status = sReport
Where RowID = tblReferences.RowID
```

The procedure then commits the table and continues the loop until the entire table is processed.

4.5.3 Tables Containing Point Objects

The sub-procedure ProcessPointTable was developed to manage the processing of tables containing point objects.

The dialog described in the previous section was for selecting an input table of point objects. A variable in the calling procedure identifies which process has been requested and titles the dialog accordingly.

The MapBasic SaveAs dialog was also used to copy the input table to a named output table as before, to protect the input table.

Prior to processing, columns for the attributes road identifier, carriageway, chainage and status report are added to the output table. The following MapBasic Statement, extracted from ProcessPointTable, was used to add the additional columns to the table:

```
Alter Table sOutputTable (Add RoadID    Char(6),
                               Cway      Char(1),
                               Chainage  Float,
                               Status    Char(50))
```

The procedure ProcessPointTable first verifies that the table contains mappable objects. If not, the process is terminated.

The Do While NOT EOT() construct described in the previous section was used to process the output able.

The method used to derive the linear referenced location of a point object in a table is similar to the interactive process described previously. However, since there is only one point object per record in a table, only one linear reference can be assigned. Therefore the procedure uses the segment object and hence carriageway type, closest to the point object.

If a point object is not within a tolerated distance of a road segment, a linear reference is not produced and an error message is assigned to the String variable sReport. Otherwise an 'OK' message is assigned.

After a record is processed, the linear reference elements and report string are added to the table. The following MapBasic statement was used to update the table for the current record:

```
Update tblPoints Set RoadId   = sRoadOut,
                     Cway     = sCwayOut,
                     Chainage = Format$(fChainage/1000, "0.###"),
                     Status   = sReport
Where RowID = tblPoints.RowID
```

The table is then committed and the procedure continues the loop until the entire table is processed.

4.6 Program Input

4.6.1 Introduction

It was necessary for the Centreline Tools upgrade to accept interactive input in both textual and spatial formats to satisfy translation requirements.

Data such as road identifiers, chainages and MapInfo table names are presented to Centreline Tools in textual format, while spatial coordinates identify locations to be translated to linear references.

The following sections discuss the implementation of interactive data input into the Centreline Tools program.

4.6.2 Textual Input

Textual input in the Centreline Tools upgrade was enabled through the use of dialog boxes.

In general terms, code used in dialog boxes in the existing version was suitable for reuse in the upgrade, albeit with varying degrees of modification.

Dialogs boxes were constructed using MapBasic 'Dialog' statements into which various combinations of 'Control' statements were placed. A number of Controls were used for a range of purposes, including:

- RadioGroup, to display a set of mutually exclusive radio buttons for selection,
- EditText, to accept input text,
- StaticText, to display textual information,
- OKButton, to accept the input and dismiss the dialog,
- CancelButton, to cancel the operation, and
- Button, for other operations activated by buttons.

Variables were used in 'Control' statements to enable the contents of dialog boxes to vary at run-time.

In the ‘Select a Carriageway’ dialog, a RadioGroup control was used to present a mutually exclusive list of carriageway options for selection. In this dialog the String array variable aCwayList was used in the 'Title' statement to populate the RadioGroup, enabling the option list to vary at run-time, based on the carriageway types involved. The following extract from the function SelectReferenceCway(), shows how the variable was used in the control:

```
Control RadioGroup
   Title From Variable aCwayList
   Position 20,47
   Value 1
   Into iListOption
```

Similarly, a dialog was used to display the options available for processing a defined section of road. The String variable sCwayString was used in the control 'Title' statement to enable a list of carriageways to be varied at run-time. The following extract from the function SelectSectionProcess() shows how the variable was used to represent a varying list of carriageway types:

```
Control StaticText
   Title sCwayString
   Position 20,20
   Width 170  Height 20
```

Handler sub-procedures were written to enable and disable dialog buttons based on the values entered into other dialog controls. Generally, one handler was written to establish the default status of the buttons in a dialog. Another handler, which is called every time a control value changes, was used to reset the button status based on input received.

Dialogs were also used to provide information and direction during interactive operations.

The MapBasic function CommandInfo(CMD_INFO_DLG_OK) was used in procedures involving dialogs to conditionally process input data. When the ‘OK’ button is pressed, the current values in the input controls are accepted, the function returns TRUE and the dialog is dismissed.

The appearance and behaviour of the dialogs used in the Centreline Tools upgrade are illustrated in Chapter 5, which describes the operation of the utility in some detail.

4.6.3 Spatial Input

Interactive spatial input into Centreline Tools involves the specification of coordinate-based point locations by mouse click events. The MapBasic functions CommandInfo(CMD_INFO_X) and CommandInfo(CMD_INFO_Y) were used in the function CaptureClickPoint() to capture the x and y coordinates of the location of a mouse click event in the map window.

The MapBasic function CreatePoint(X, Y) was used to create a point object at the captured coordinates, which could then be assigned to an object type variable for use elsewhere in the program. The following extract from the function CaptureClickPoint() demonstrates the use of the MapBasic functions to create a point object:

```
fClickX = CommandInfo(CMD_INFO_X)
fClickY = CommandInfo(CMD_INFO_Y)

'--- create a point object at the captured coordinates
oMouseClickPoint = CreatePoint(fClickX, fClickY)
```

The function CaptureClickPoint() was used to capture the spatial locations for which linear references were required, and the definition of road sections by beginning and end spatial locations.

4.7 System Testing

4.7.1 Introduction

Testing was an important facet of the development of the Centreline Tools upgrade and several strategies were employed. Development testing was carried out during the design and coding phases of the project. Operational testing involved the limited release of completed development versions to be tested under operational conditions by regular users of the system.

4.7.2 Development Testing

The sub-procedure MakeDebugTable was written to create a MapInfo table in which to store debugging information. The structure and behaviour of spatial objects constructed by the program could then be examined and manipulated in a MapInfo environment. Figure 4.16 shows an example of the debugging environment containing the spatial objects used to determine the linear reference at a given spatial location.

Additional statements were written into the source code to display debugging information such as variable values during program execution. These sections of code were enclosed by condition statements controlled by the value of the Logical (boolean) data type program constant CT_DEBUG_ON, which was defined in the definition file. The display of debugging information could then be turned off for development version releases simply by setting the value of CT_DEBUG_ON to FALSE in the definition file prior to compilation. This strategy also enabled sections of debugging code to be left in place for possible use in the future. The following extract from the definition file shows the definition statements for the debugging constants:

```
DEFINE CT_DEBUG_ON        FALSE   'TRUE  '- set T for development
DEFINE CT_DEBUG_TAB       "Debug"        '- debugging table name
```

Figure 4.17 demonstrates the use of debugging code to display the incrementing value of a node number variable, and chainage calculation values during a program loop structure.

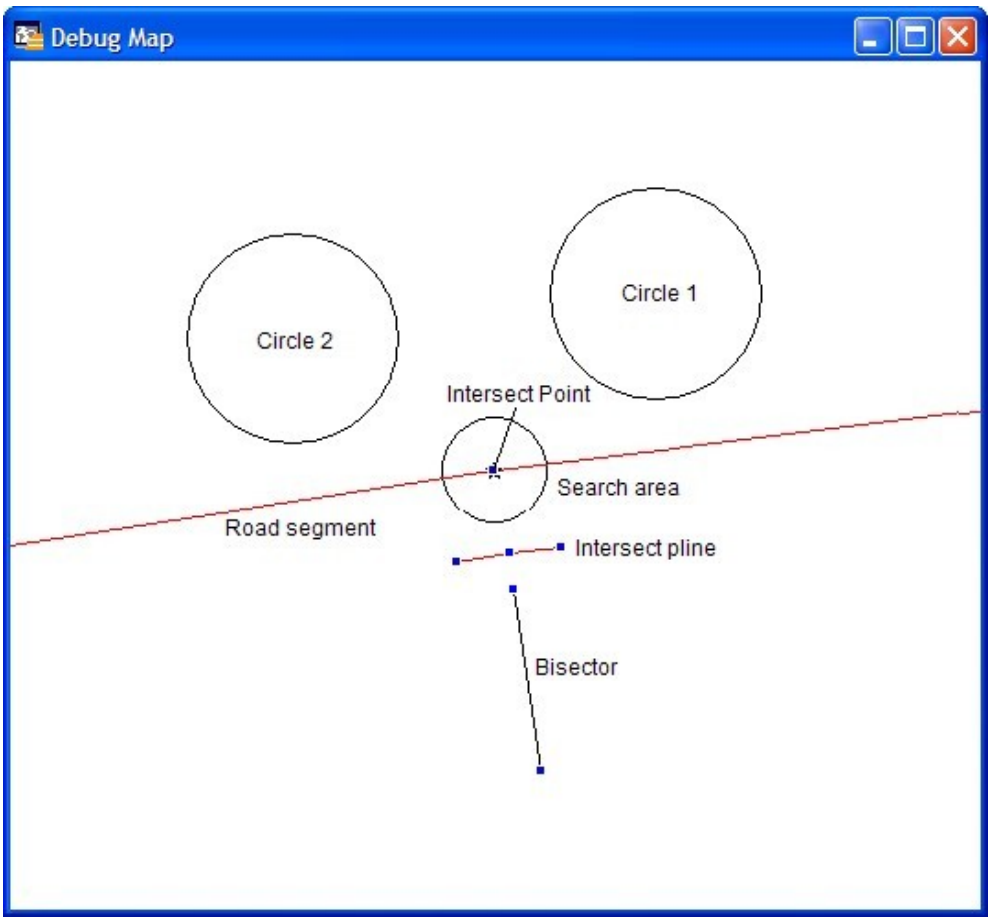


Figure 4.16 – Spatial Debugging Environment

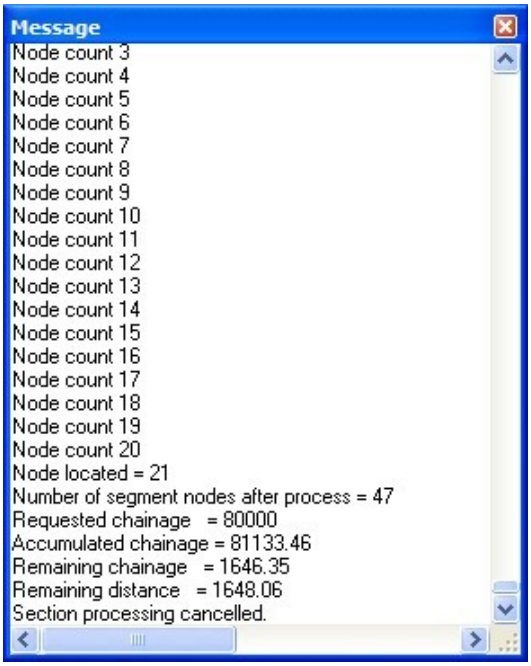


Figure 4.17 – Node Number Variable Debugging Output

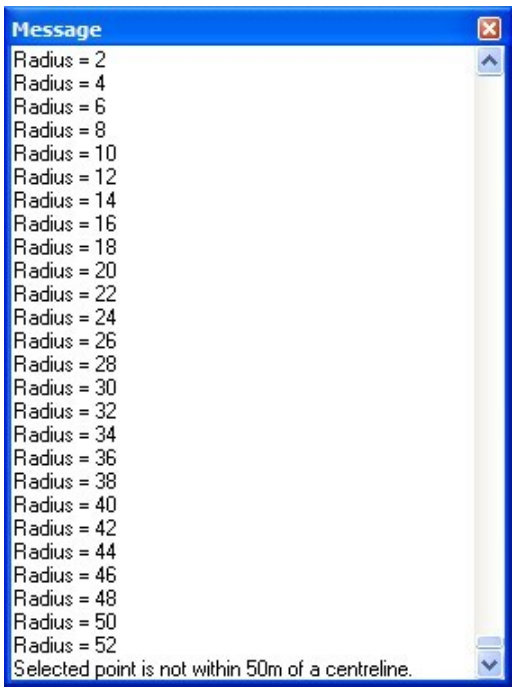


Figure 4.18 – Expanding Search Area Debugging Output

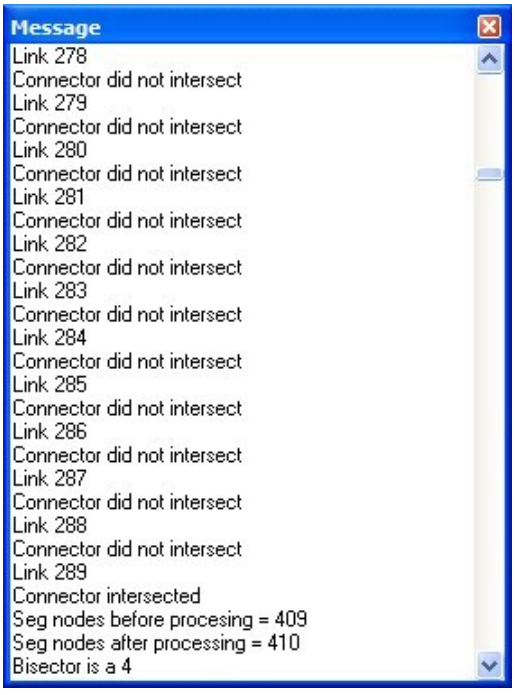


Figure 4.19 – Link Intersection Debugging Output

Another example of the use of debugging code is shown in Figure 4.18, which shows the increasing value of the radius variable used to incrementally expand a search area in search of a road segment.

Figure 4.19 illustrates another example of debugging code which displays variable values during a traverse through the links of a segment object in search of an intersection condition.

Program outputs from interactive processes were assessed in relation to expected outcomes and examined manually using other MapInfo tools. Outputs were also compared against the results of similar operations performed by the existing version.

In order to accommodate comparative testing between development versions of the upgrade and the existing version, some changes were made to system file names. The name of the MapBasic executable file for the upgrade was changed to CT_Main.MBX from CL_Main.MBX, and the name of the temporary system data table was also changed to CT_Data.TAB from CL_Data.TAB. These changes allowed the upgrade versions to run in the same instance of MapInfo as the existing version without conflicts.

Specifically designed input data was also used for development testing and served several purposes. Firstly, prepared data was used to ensure that the majority of logical branches in the program were executed.

Secondly, data was designed to test the program's operation with the entire spectrum of road network complexity.

Finally, prepared input data enabled testing of the program's response to potentially problematic scenarios, including:

- incorrect road identifiers,
- chainages known to exceed the extents of a road,
- road/carriageway combinations known not to exist,
- linear references known to occur at RPs,
- section start and end points occurring on different roads,
- section start and end chainages occurring at RPs,
- section start and end chainages both exceeding extents, and
- sections defined against gazettal direction.

Appendix E illustrates the use of specifically prepared input data for system testing. The attributes of the input table contain comments describing known anomalies in the test data. The 'Status' column shows the report generated for each record during processing. The Centreline Tools environment shows some of the spatial objects produced from the input data during segmentation.

4.7.3 Operational Testing

Several versions of the Centreline Tools upgrade were released to the test group for operational testing during the project. Details of the releases are listed in Table 4.1.

Centreline Tools Version	Release Date
Development Version 5.0.Alpha	6th June 2008
Development Version 5.0.Beta	18th July 2008
Development Version 5.0.Beta.2	19th August 2008
Production Version 5.0	23rd September 2008

Table 4.1 – Versions of Centreline Tools Released

The Alpha version mandated the inclusion of a carriageway code element in the dialog used to set the parameters for dynamic segmentation. However, feedback suggested that mandating the selection of a carriageway code element complicated the process unnecessarily. It was rightly pointed out that the majority of the state does not have roads made up of multiple carriageways, so in many cases a carriageway code is not required (Appleman, A 2008, pers. comm., 20 June).

As a result of the feedback provided, the dynamic segmentation process in relation to carriageway code was modified. The mandated requirement for a carriageway code column was removed and is now used only if selected. Otherwise, each linear reference is mapped to a through carriageway, or the actual carriageways on which it occurs (through carriageways are 1, 2 and 3). Removing the necessity for the inclusion of a carriageway code also simplified the preparation of an input file to be segmented.

The Version 5.0.Beta release of Centreline Tools included support for extracting (copying) sections of roads defined by start and end points and linear references. The modifications required for the entry of carriageway codes were also incorporated into this release.

The additional section definition toolbar used in the existing version was absorbed into the Centreline Tools toolbar because of an issue with the focus of windows changing away from MapInfo on closing the section toolbar. This behaviour was considered to be unacceptable from a user perspective, and as the cause was not immediately apparent in the Main Roads environment, a decision was made to avoid the problem by combining the section definition buttons into the Centreline Tools toolbar.

Soon after the release of the 5.0.Beta version, it was reported that the 'Process' button was not activating correctly in the dialog used to set dynamic segmentation parameters (Martin, AL 2008, pers. comm., 12 Aug). This problem was traced to the logic of the procedure VerifySelectColumns, which failed to activate the ‘Process’ button if the road identifier and carriageway columns were both selected. It appears that this procedure was incorrectly modified while removing the mandatory requirement for a carriageway code in the previous release. The sub-procedure VerifySelectColumns was rewritten to ensure the ‘Process’ button activated correctly.

Following these modifications, Centreline Tools was re-released on 19th August as Version 5.0.Beta.2.

After an additional period of operational testing, during which no problems were reported, a final version was released into production on 23rd September as Centreline Tools Version 5.0.

4.8 Conclusions

During the system development process, several methodologies were pursued unsuccessful for a variety of reasons, principally the unreliability of several MapBasic functions in relation to intersecting objects.

Initially, the MapBasic function ConnectObjects(o*bject1*, o*bject2*, min) was used to create a node at the location on a segment object that is closest to a given point. This function returns a two-point polyline object representing the shortest distance between the two objects, with one end on o*bject1* (the given point) and the other on o*bject2*, (the centreline segment). The method involved creating and finding a node in the segment (o*bject2*) at the intersection point with the shortest distance polyline returned by the ConnectObjects() function. However, when searching for the intersection point between the two objects, run-time errors were often encountered because the polyline and the segment did not, according to MapInfo, intersect. Although the use of ConnectObjects() was the preferred approach, it was abandoned because it was not considered to be sufficiently reliable for use in the Centreline Tools upgrade.

Other errors were occasionally encountered with object intersections, possible because the single-precision nature of MapInfo is not sufficiently accurate to place nodes precisely into objects.

A similar issue arose in the function FindPointSegments(), which is used to select the road centreline segments that intersect with an expanding circular search area. This procedure contains a select statement with a spatial intersection condition clause, which when assessed as TRUE, causes the search loop to terminate. However, during further processing, a run-time error occasionally occurred claiming that the objects did not intersect. Introducing an additional test to verify that an intersection condition did exist solved this problem.

Unexpected 'Line' object types in *sc_roads* caused run-time errors during system development. Some MapBasic procedures involving polylines fail if applied to a line object, including those used to add nodes and return node numbers. Therefore, the sub-procedure VerifyPolyline was included to test object types prior to the use of these functions, and convert lines to polylines if they were encountered.

CHAPTER 5
RESULTS

5.1 Introduction

When the Centreline Tools program is launched, a toolbar is created and placed into the MapInfo environment. This toolbar provides the interface through which the operations performed by the utility are accessed. The toolbar from the Centreline Tools upgrade is shown in Figure 5.1, together with a brief description of each button.

The remainder of this chapter describes the operation performed by each tool button available in the production version of the upgrade, Centreline Tools Version 5.0.

5.2 Linear Reference at a Point

Selecting the 'Linear reference at point' tool button (see Figure 5.1) enables the determination and display of linear references at selected spatial locations on the road network.

This button is a push-button and once selected, it remains active until another tool button is selected. When the push-button is depressed, the cursor changes to a crosshair when moved into the map window. The spatial location at which a linear reference is required, is specified by positioning the cursor at the location, and clicking the mouse button.

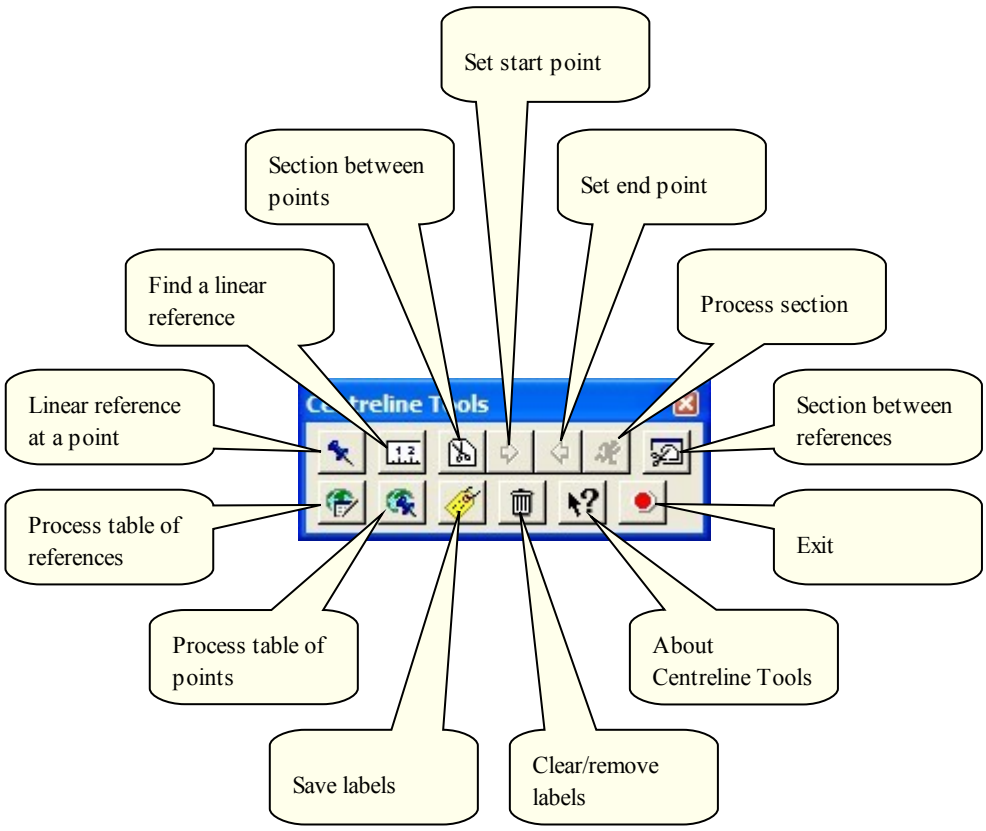


Figure 5.1 – Centreline Tools V5.0 Toolbar

When a location is selected, a point object representing the position on the road closest to the selected location, and a set of attributes are written to the Centreline Tools data table called *CT_Data*. A label expression is constructed from the road identifier, carriageway and chainage attributes and is also written to the table *CT_Data.* The point object is displayed in the map window labelled with the constructed expression.

Appendix F illustrates the Centreline Tools environment following the use of the 'Linear reference at point' tool button to obtain a number of linear references. The attributes of *CT_Data* generated during this activity are displayed in Table 5.1. The 'Item' attribute identifies the operation performed.

CT_Data Browser

Centreline	Item	RoadID	Carriageway	StartTDist	EndTDist	Length	LabelText
sc_roads	Selected point	10H	1	52.753	52.753	0	Road 10H - Cw 1□TDist 52.753km
sc_roads	Selected point	855	1	15.662	15.662	0	Road 855 - Cw 1□TDist 15.662km
sc_roads	Selected point	10H	1	39.157	39.157	0	Road 10H - Cw 1□TDist 39.157km
sc_roads	Selected point	536	1	20.092	20.092	0	Road 536 - Cw 1□TDist 20.092km
sc_roads	Selected point	532	1	48.745	48.745	0	Road 532 - Cw 1□TDist 48.745km
sc_roads	Selected point	534	1	4.557	4.557	0	Road 534 - Cw 1□TDist 4.557km
sc_roads	Selected point	33B	1	71.376	71.376	0	Road 33B - Cw 1□TDist 71.376km
sc_roads	Selected point	10H	1	16.87	16.87	0	Road 10H - Cw 1□TDist 16.870km
sc_roads	Selected point	857	1	6.809	6.809	0	Road 857 - Cw 1□TDist 6.809km
sc_roads	Selected point	517	1	21.998	21.998	0	Road 517 - Cw 1□TDist 21.998km
sc_roads	Selected point	33B	1	49.414	49.414	0	Road 33B - Cw 1□TDist 49.414km
sc_roads	Selected point	8554	1	7.755	7.755	0	Road 8554 - Cw 1□TDist 7.755km

Table 5.1 – *CT_Data* Attributes for Specified Spatial Locations

5.3 Find a Linear Reference

This operation determines the spatial coordinates of the location at which a specified linear reference occurs, and centres the map window at the coordinates.

After selecting the 'Find a linear reference' button (see Figure 5.1), the dialog displayed in Figure 5.2 is presented for the entry of the desired linear reference. The entered chainage can be in metres or kilometres, but the appropriate 'units' must be specified.

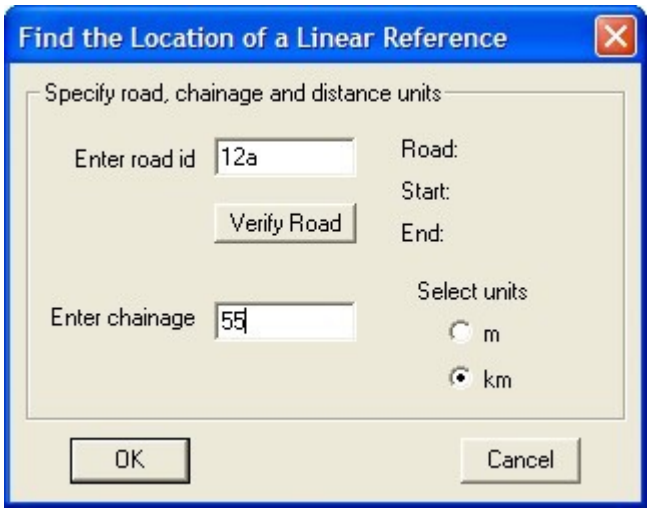


Figure 5.2 – Entry Dialog for Linear References

The ‘Verify Road’ button in the entry dialog provides the option to display the minimum and maximum chainage of the entered road identifier. Clicking the ‘Verify Road’ button displays the road extents, as shown in Figure 5.3.

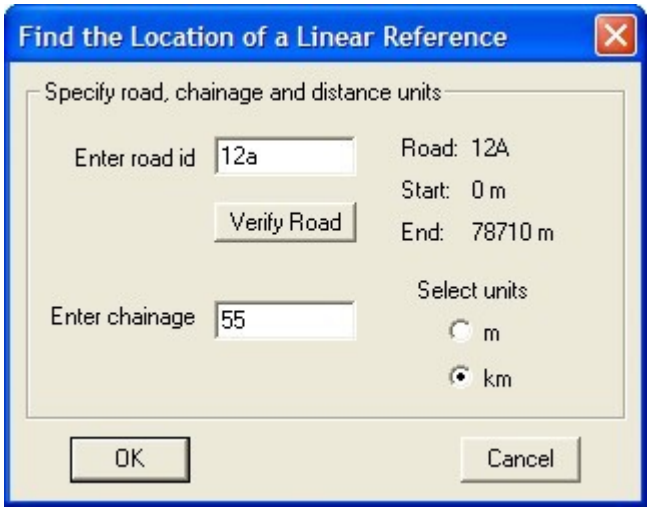


Figure 5.3 – Verification of Road Extents

Several validations are carried out on an entered linear reference. If the road identifier does not represent a valid road, or the chainage is beyond the extents of the road, the user is advised and the dialog remains in place. Error messages generated for invalid entries are displayed in Figure 5.4.

Figure 5.4 – Linear Reference Entry Error Messages

After a valid linear reference has been entered, clicking the OK button dismisses the dialog and Centreline Tools finds the coordinates of the requested location and creates a point object. As in the previous operation the point object and relevant attributes are added to the table *CT_Data*. The map window is then centred at the coordinates of the linear reference and the point object and a label similar to that in the previous section are displayed.

If the requested linear reference occurs on multiple carriageways, a dialog is presented from which to select a carriageway option for processing. Figure 5.5 is the dialog produced by the linear reference requested on road 12A shown in Figure 5.3. This dialog is not presented if the requested linear reference occurs on one carriageway only.

Figure 5.5 – Selection Dialog for Carriageway Options

If the 'All carriageways' option is selected, a record for each carriageway is added to *CT_Data* and multiple points and labels are displayed.

Table 5.2 contains the attributes generated after the 'All carriageways' option selection shown in Figure 5.5. Comparison with the attributes in Table 5.1 shows that the 'Item' column identifies the operation performed as 'Specified linear reference'.

CT_Data Browser

Centreline	Item	RoadID	Carriageway	StartTDist	EndTDist	Length	LabelText
sc_roads	Specified linear reference	12A	2	55	55	0	Road 12A - Cw 2□TDist 55.00km
sc_roads	Specified linear reference	12A	3	55	55	0	Road 12A - Cw 3□TDist 55.00km
sc_roads	Specified linear reference	12A	A	55	55	0	Road 12A - Cw A□TDist 55.00km
sc_roads	Specified linear reference	12A	Q	55	55	0	Road 12A - Cw Q□TDist 55.00km

Table 5.2 – *CT_Data* Attributes for Specified Linear References

Appendix G illustrates the Centreline Tools environment after using the 'Find a linear reference' tool button to locate the linear reference shown in Figure 5.3, with the 'All carriageways'option as shown in Figure 5.5.

5.4 Process Table of Linear References

Selecting the ‘Process table of references’ tool button (see Figure 5.1) triggers the dynamic segmentation process, which is also referred to within MR as ‘geocoding’.

When the ‘Process table of references’ tool button is selected, a dialog is presented for the selection of an input table containing linear references to be dynamically segmented. An open table can be selected, or a table can be located and selected in a browser. Figure 5.6 shows the input dialog with the table 'Access16A' selected. This table contains the locations of accesses to a state-controlled road in linear reference format.

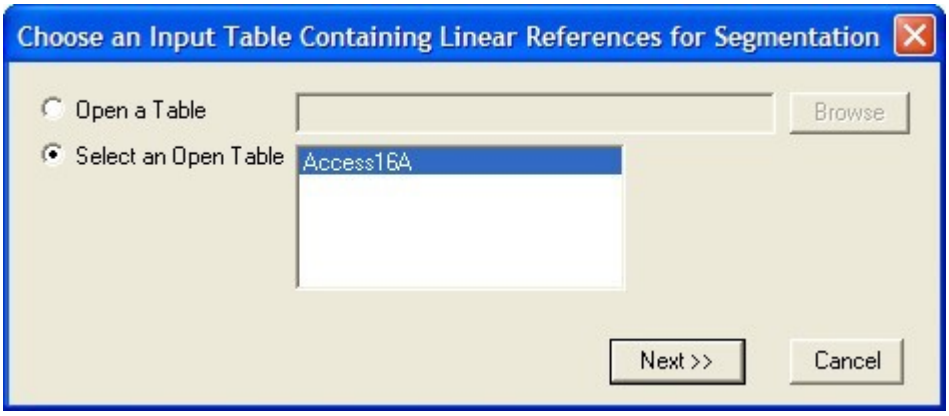


Figure 5.6 – Selection Dialog for an Input Table of Linear References

After the input table is selected, a MapInfo SaveAs Dialog is presented to save a copy of the input table as an output table. The spatial objects created during processing are added to the output table, leaving the input table unmodified. In the Figure 5.7 example, the input table ‘Access16A’ is saved as the output table ‘Access16A_Mapped’. When the output table is created an additional column is added for a brief status report on the result of the segmentation process for each record.

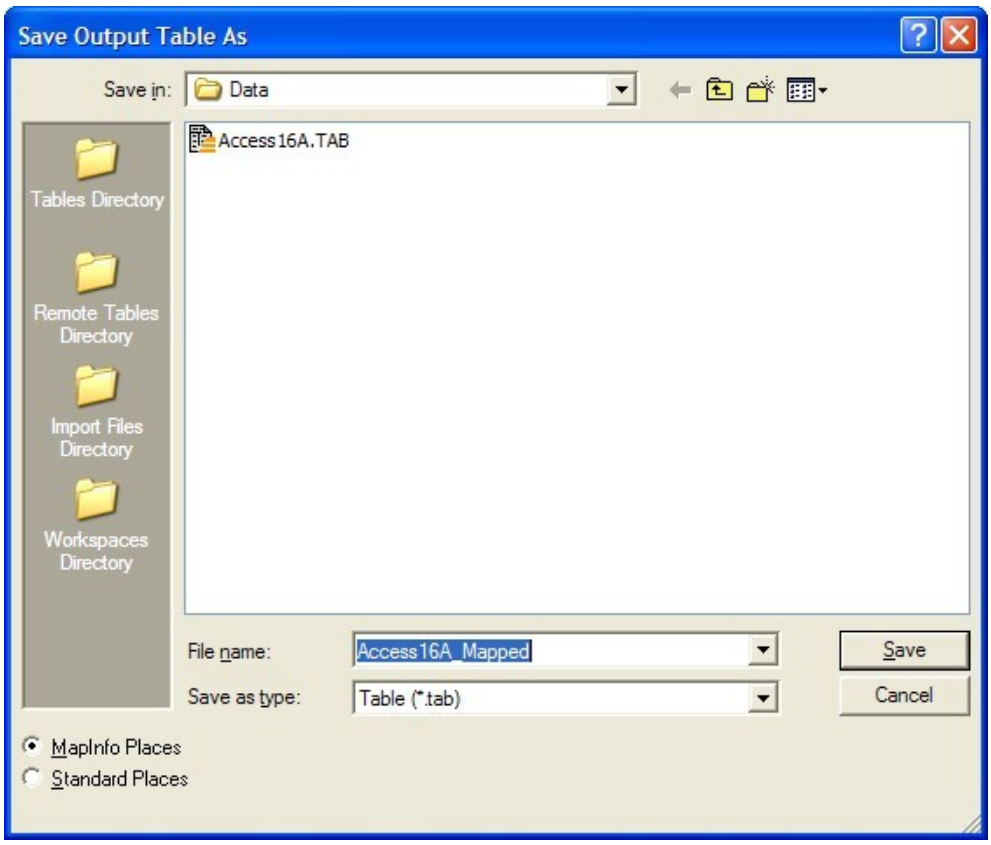


Figure 5.7 – Selection Dialog for an Output Linear Reference Table

Once the output table has been created, the dynamic segmentation parameters must be specified in the dialog shown in Figure 5.8. Firstly, the format of the linear references is selected. Centreline Tools can segment data in the three formats described at the top of Figure 5.8. In the Figure 5.8 example, the 'single chainage value' format is selected, which results in the creation of point objects. Polyline objects can also be created for linear sections in the formats of start and end chainages, and start chainage and section length.

Next, the table columns containing the values representing the selected linear reference elements must be specified. The column known to contain the road identifier is selected from the list under 'Road identifier column'. The column containing the carriageway code can also be specified if required.

Following the specification of the single chainage format, the 'Select column for a single chainage value' section of the dialog is activated for entry, as shown in Figure 5.8. The name of the column known to contain the single chainage value is selected from the drop-down list in this section. Finally, the chainage units are specified.

The ‘Process’ button on the dialog is activated only after the columns required for the specified linear reference format have been selected. Table processing commences when the ‘Process’ button is clicked.

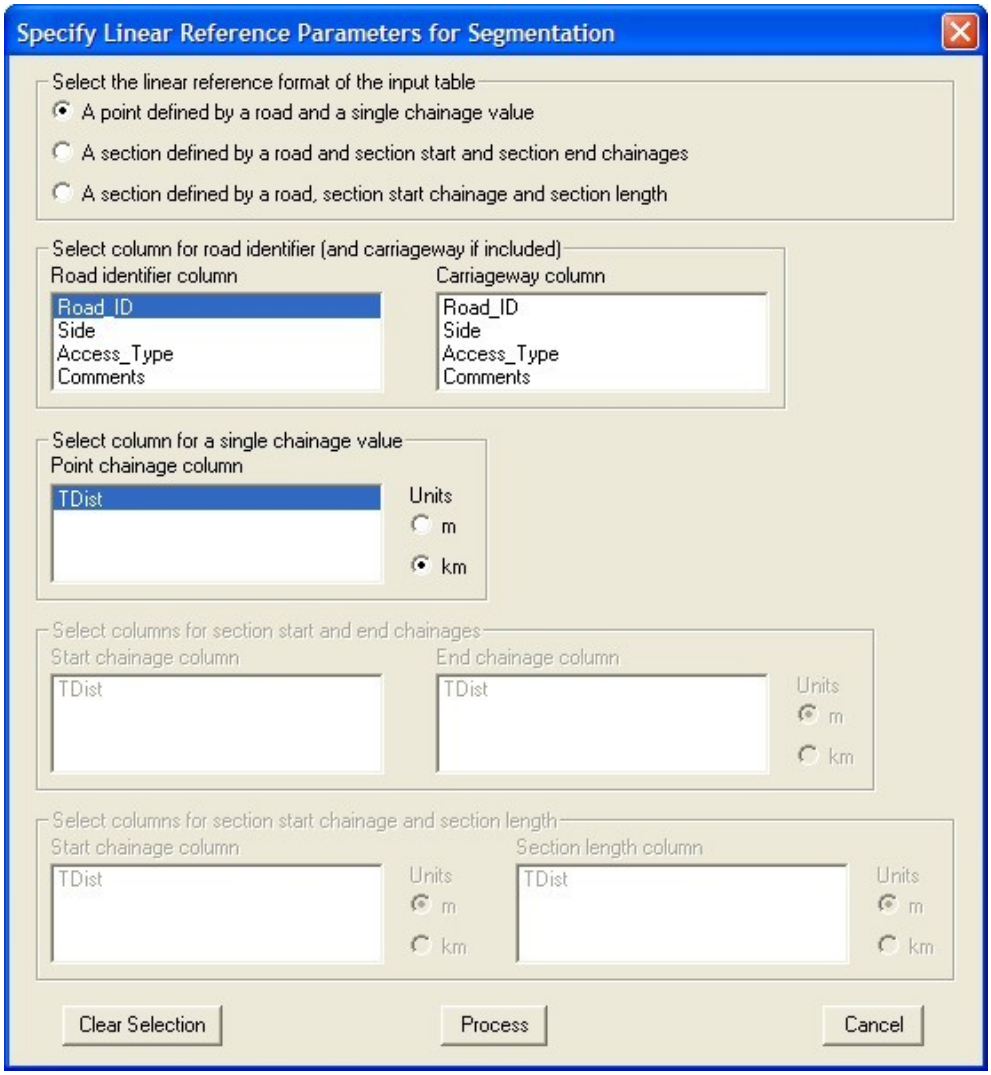


Figure 5.8 – Dynamic Segmentation Configuration Dialog

During processing, tests are conducted to validate each linear reference. Road identifier, road extents and the order of linear references are verified. Chainages that are beyond the road extents are reset to the end of the road and noted in the status report. Records containing invalid road identifiers cannot be processed. The status value is also displayed in the Message window during processing.

Appendix H.A illustrates the Centreline Tools environment following dynamic segmentation. To better demonstrate the outcome of the process, the point objects of the

table ‘Access16A_Mapped’ created during segmentation, have been thematically mapped by the attribute 'Access_Type', and labelled with the chainage attribute ‘TDist’.

Appendix H.B compares the attributes of the input table ‘Access16A’ with that of the output table ‘Access16A_Mapped’, after processing. It can be seen from the ‘Status’ column that all the records in the table were correctly geocoded.

5.5 Process Table of Points

Selection of the ‘Process table of points’ tool button (see Figure 5.1) enables the processing of a MapInfo input table containing point objects. In this process, a linear reference is generated for each point object in the input table. This process is often referred to in MR as ‘reverse geocoding’.

After the tool button is selected, the dialog shown in Figure 5.9 is presented for selection of a MapInfo input table. An open table can be selected, or a table can be located and selected in a browser. The example shown in Figure 5.9 shows the open table 'FloraSites' selected as the input table. This table contains point objects representing site data collected in the road corridor using a GPS device.

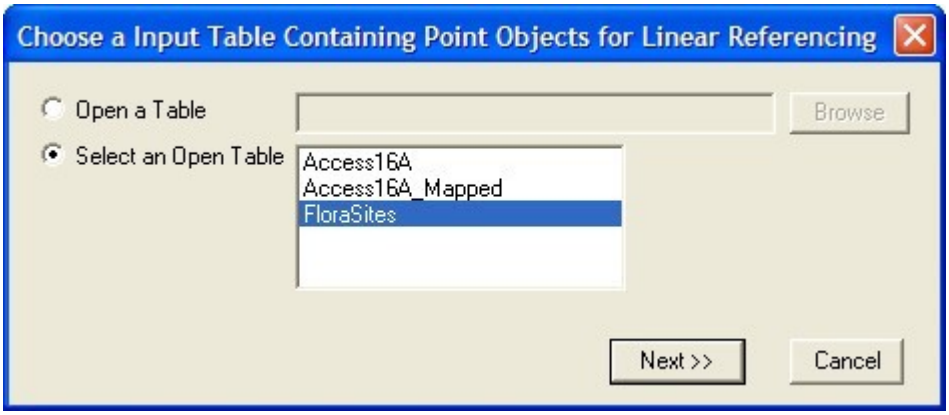


Figure 5.9 – Selection Dialog for an Input Table of Point Objects

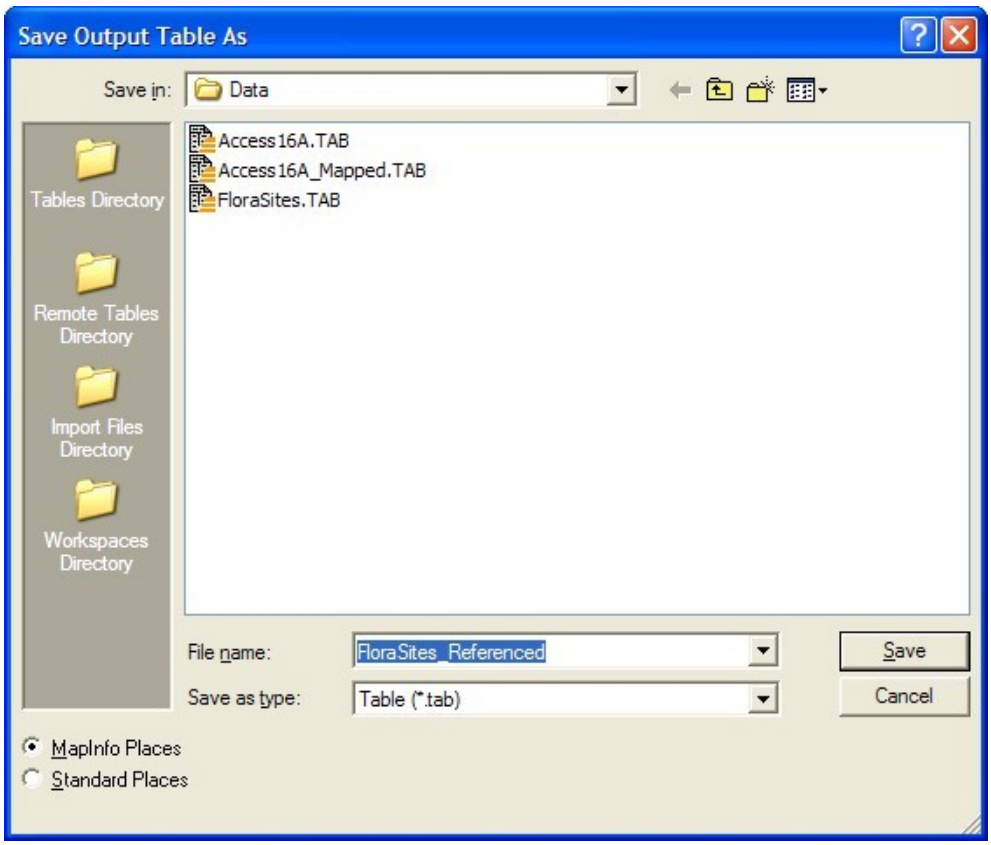


Figure 5.10 – Selection Dialog for an Output Points Table

An output table is created in the same way as the previous operation to protect the input table. In the Figure 5.10 example, the input table is copied to an output table called 'FloraSites_Referenced'. During the creation of the output table, additional columns are added for the linear reference elements and a brief status report on the referencing process.

Centreline Tools then processes the output table, generating a linear reference for each point object. The linear reference consists of a road identifier, carriageway code and chainage. The status column is populated on the basis of the outcome of the referencing process.

Appendix I.A displays the attributes of the input table 'FloraSites' and the output table 'FloraSites_Referenced' after processing, and illustrates the additional columns populated with the linear reference elements.

It can also be seen in Appendix I.A that a linear reference was not resolved for two point records, which have a 'Status' value of 'Point is not near a road segment'. In these

cases the point objects were not within a tolerated distance of a road segment object, which is currently preset to 50 metres.

Appendix I.B illustrates the Centreline Tools environment after using the ‘Process table of points’ button to 'reverse geocode' a table of point objects. As in the previous section, the display has been enhanced to better demonstrate the operation. The processed data has been thematically mapped by ‘Status’ and labelled with the linear reference generated. The unsuccessfully referenced points are labelled in red.

5.6 Section Between Points

The purpose of the operation enabled by the ‘Section between points’ tool button (see Figure 5.1) is to create spatial objects representing sections of road that occur between two specified geographic locations.

Defining a road section between coordinates involves specifying start and end points in a similar way to requesting a linear reference at a point, described above in Section 5.2.

Selecting the ‘Section between points’ tool button activates the ‘Set start point’ tool button (see Figure 5.1). When this button is activated, the cursor becomes a crosshair in the map window, enabling a section start point to be created. The start point is displayed and labelled with the identifier of the road in which the start point occurs.

Successfully specifying the start point enables the ‘Set end point’ tool button (see Figure 5.1). The section end point is specified in the same way as the start point, and is similarly displayed. Figure 5.11 shows a map window extract containing a section definition between start and end coordinates.

Centreline Tools validates point selections. Firstly, the end point must be on the same road as the start point. Secondly, the section must be defined in gazettal direction (in the direction of increasing chainage). If either of these conditions is not met, an error message is displayed and the section ends must be respecified before continuing. Messages displayed by Centreline Tools for invalid section end point definitions are displayed in Figure 5.12.

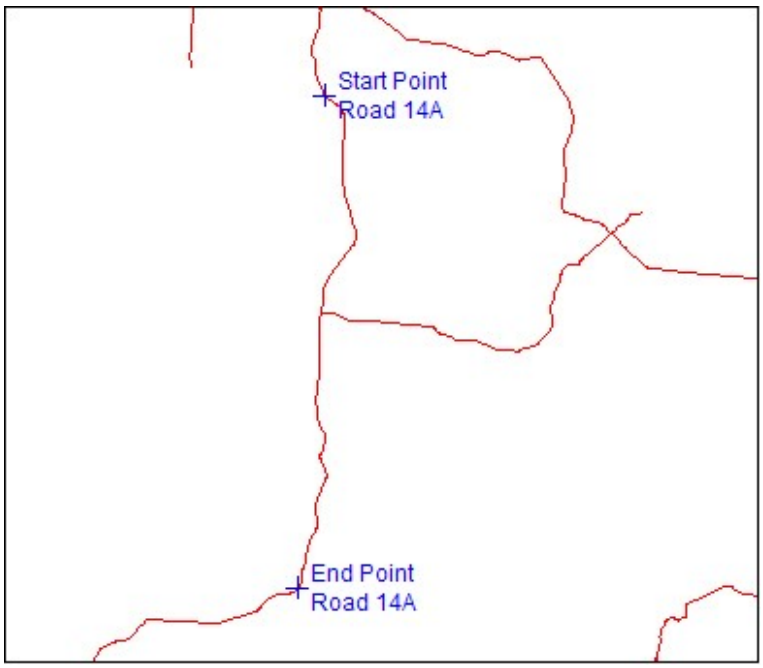


Figure 5.11 – Defining a Section Between Coordinates

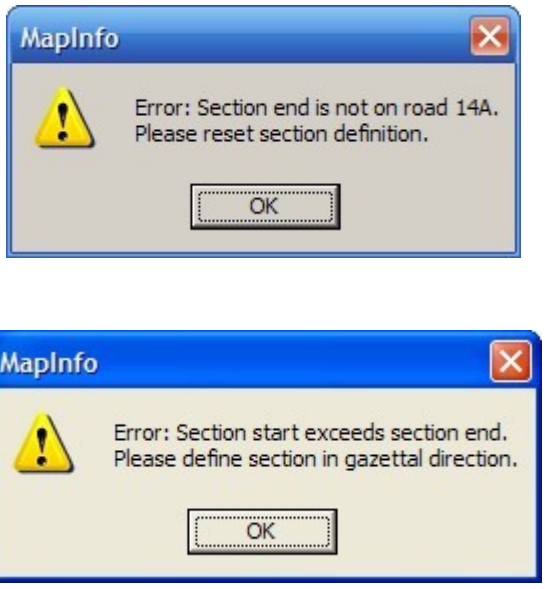


Figure 5.12 – Error Messages for a Section Defined by Coordinates

Only when the section has been correctly defined, is the 'Process section' tool button enabled (see Figure 5.1). After activating the 'Process section' tool button, the defined section elements are displayed in the map window and a dialog such as the example shown in Figure 5.13 presents the available options for preparation of the section elements. The section elements referred to in the dialog are the segment objects or parts

thereof (portions) that collectively make up the defined section, and can be presented in a number of ways.

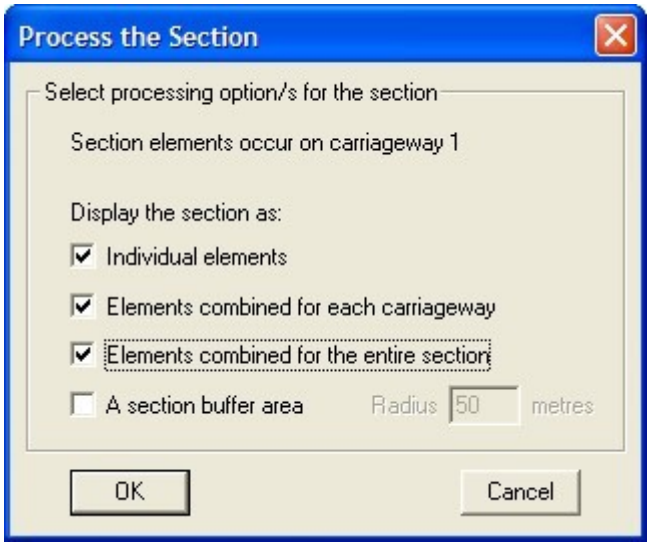


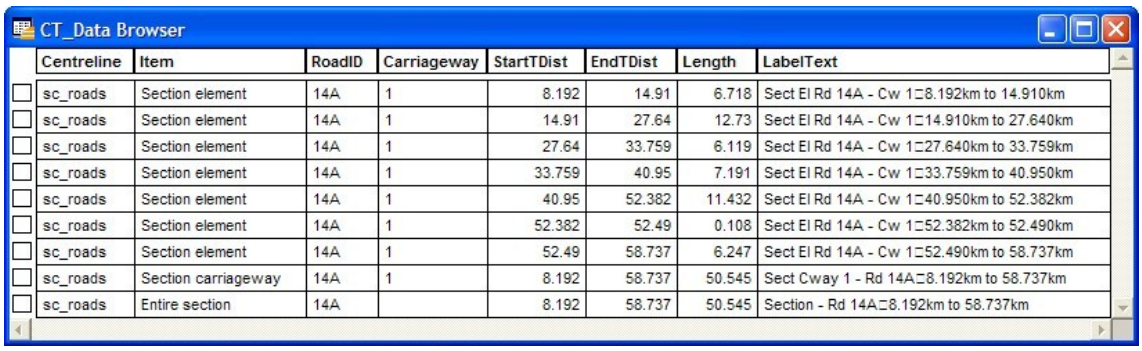
Figure 5.13 – Processing Options for a Section on a Single Carriageway

The section elements can be displayed as individual elements, combined into a single object for each carriageway, or combined into a single object representing the entire section. An option to create a buffer object around the entire section, with a specified radius, is also available. The options are not mutually exclusive and any combination of the options can be selected.

Once the desired options have been selected, the section objects are created and written to the table *CT_Data*, together with the relative attributes. The constructed section objects and their labels are then displayed in the map window.

Appendix J shows the Centreline Tools environment following the definition of the section on Road 14A by end point coordinates. The section objects representing the individual elements, combined carriageway and entire section options selected in Figure 5.13 are displayed.

Table 5.3 shows the attributes written to the table *CT_Data* for the section display options selected in Figure 5.13.



Centreline	Item	RoadID	Carriageway	StartTDist	EndTDist	Length	LabelText
sc_roads	Section element	14A	1	8.192	14.91	6.718	Sect El Rd 14A - Cw 1⊐8.192km to 14.910km
sc_roads	Section element	14A	1	14.91	27.64	12.73	Sect El Rd 14A - Cw 1⊐14.910km to 27.640km
sc_roads	Section element	14A	1	27.64	33.759	6.119	Sect El Rd 14A - Cw 1⊐27.640km to 33.759km
sc_roads	Section element	14A	1	33.759	40.95	7.191	Sect El Rd 14A - Cw 1⊐33.759km to 40.950km
sc_roads	Section element	14A	1	40.95	52.382	11.432	Sect El Rd 14A - Cw 1⊐40.950km to 52.382km
sc_roads	Section element	14A	1	52.382	52.49	0.108	Sect El Rd 14A - Cw 1⊐52.382km to 52.490km
sc_roads	Section element	14A	1	52.49	58.737	6.247	Sect El Rd 14A - Cw 1⊐52.490km to 58.737km
sc_roads	Section carriageway	14A	1	8.192	58.737	50.545	Sect Cway 1 - Rd 14A⊐8.192km to 58.737km
sc_roads	Entire section	14A		8.192	58.737	50.545	Section - Rd 14A⊐8.192km to 58.737km

Table 5.3 – *CT_Data* Attributes for a Section Defined by Coordinates

5.7 Section Between Linear References

The operation performed by the ‘Section between references’ tool button (see Figure 5.1) is similar to that of the ‘Section between points’ button described in the previous section. However, the road section is defined by the entry of section start and end linear references.

Selecting the ‘Section between references’ tool button immediately opens a dialog in which to define a section of road. This dialog, shown in Figure 5.14, is similar to that used for finding a linear reference described in Section 5.3, but two chainage values must be entered to specify the start and end of the road section.

The ‘Verify Road’ button optionally returns the extents of the entered road identifier as previously described. The entered road identifier is validated as previously described in Section 5.3. Entered start and end chainage values are verified also. Invalid entries are rejected, and error messages shown in Figure 5.15 are displayed.

The road section must be defined in gazettal direction. Should the entered section end chainage be less than the section start, the message also shown in Figure 5.15 is displayed, and the section must be redefined.

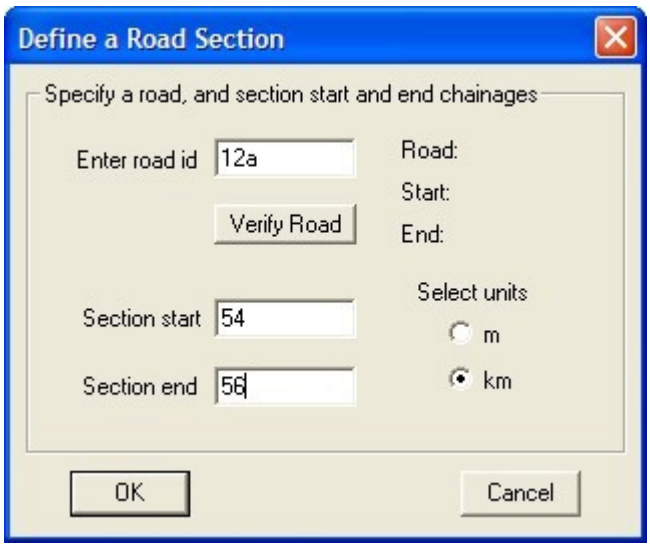


Figure 5.14 – Entry Dialog for Defining a Section by Linear References

Figure 5.15 – Error Messages for a Section Defined by Linear References

Once a valid road section has been defined, the process follows that described in the previous section. The example processing option selection dialog shown in Figure 5.16 represents a section defined by linear references on Road 12A, which contains multiple carriageways. In this case, elements combined for each carriageway and a section buffer set to a radius of 100m have been selected. The objects representing the selected options are then created and written to the table *CT_Data*, together with the relative attributes. The constructed section objects and their labels are then displayed in the map window.

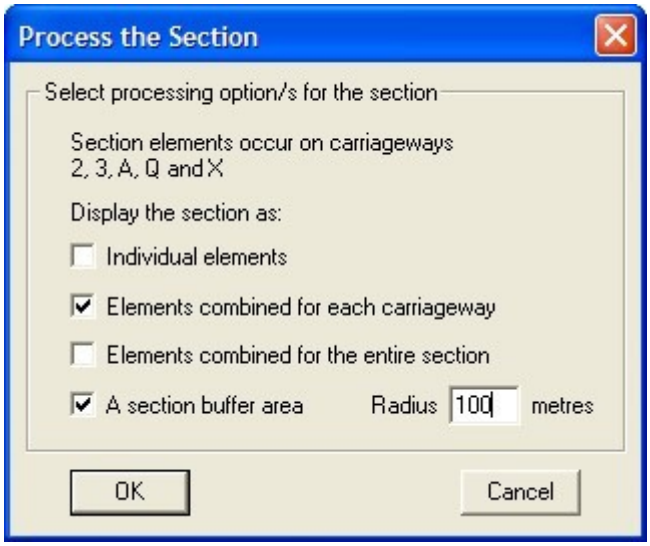


Figure 5.16 – Processing Options for a Section on Multiple Carriageways

Appendix K shows the Centreline Tools environment following the definition of the section between two linear references on Road 12A. The options selected in Figure 5.16 are displayed.

The attributes written to the table *CT_Data* for the Road 12A section options are shown in Table 5.4.

CT_Data Browser

Centreline	Item	RoadID	Carriageway	StartTDist	EndTDist	Length	LabelText
sc_roads	Section carriageway	12A	2	54	56	2	Sect Cway 2 - Rd 12A□54.00km to 56.00km
sc_roads	Section carriageway	12A	3	54	56	2	Sect Cway 3 - Rd 12A□54.00km to 56.00km
sc_roads	Section carriageway	12A	A	54.8	55.915	1.115	Sect Cway A - Rd 12A□54.80km to 55.915km
sc_roads	Section carriageway	12A	Q	54.76	55.87	1.11	Sect Cway Q - Rd 12A□54.760km to 55.870km
sc_roads	Section carriageway	12A	X	55.09	55.31	0.22	Sect Cway X - Rd 12A□55.090km to 55.310km
sc_roads	Section buffer	12A		54	56	2	Sect Buffer - Rd 12A□54.00km to 56.00km

Table 5.4 – *CT_Data* Attributes for a Section Defined by Linear References

5.8 Save Labels

The table *CT_Data* is a temporary table and is not saved when Centreline Tools is closed. The 'Save labels' tool button (see Figure 5.1) provides the option to save the current contents of *CT_Data* to a permanent table. When the operation is performed, the dialog shown in Figure 5.17 is presented in which to enter a name for the saved table. The name 'CT_lables' appears initially as a default option. After being copied to the new table, *CT_Data* is closed, removing the objects and labels currently displayed in the map window. The table *CT_Data* is recreated next time Centreline Tools performs an operation.

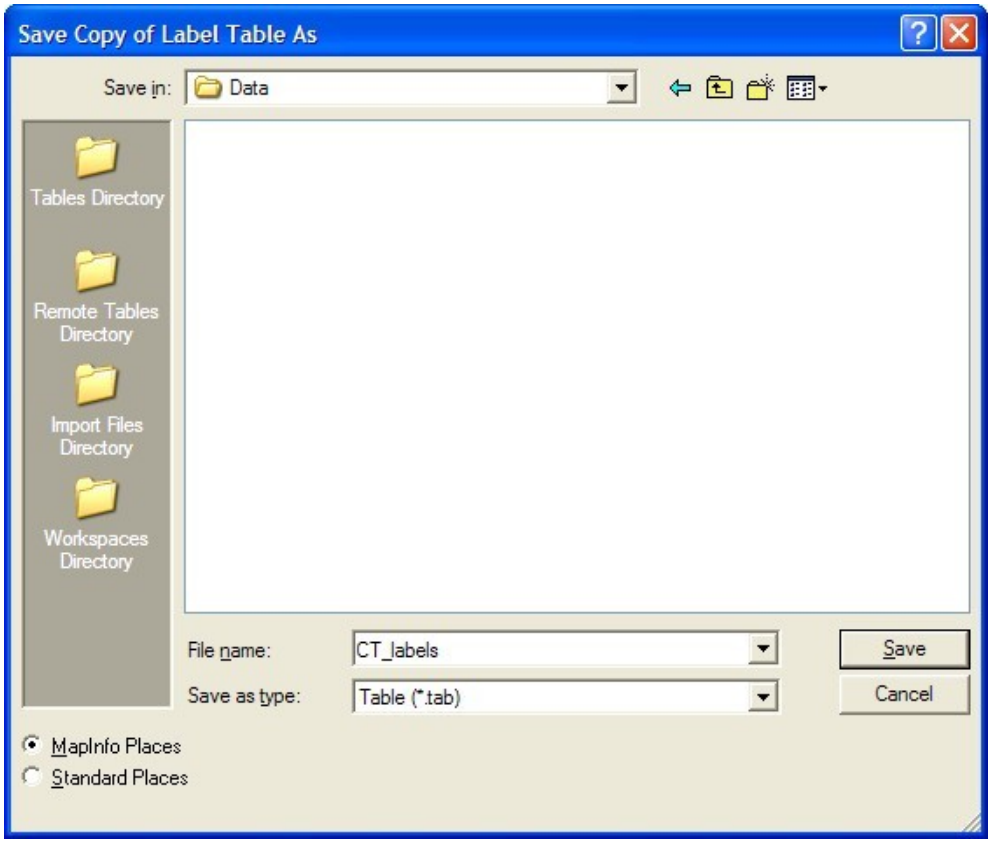


Figure 5.17 – Selection Dialog to Save the Centreline Tools Table *CT_Data*

5.9 Clear/Remove Labels

The operation performed by the 'Clear/remove labels' tool button (see Figure 5.1) is twofold. Firstly, the table *CT_Data* is closed, automatically removing any related objects and labels currently displayed in the map window.

Secondly, the tool buttons are reset to their default states, removing any persisting section specification point definitions.

5.10 About Centreline Tools

Information about the program is obtained by selecting the 'About Centreline Tools' tool button (see Figure 5.1). The information provided includes the version number and release date, which are particularly important when providing client support. Figure 5.18 shows the 'About Centreline Tools' dialog box produced by the production release of the Centreline Tools upgrade developed for this project.

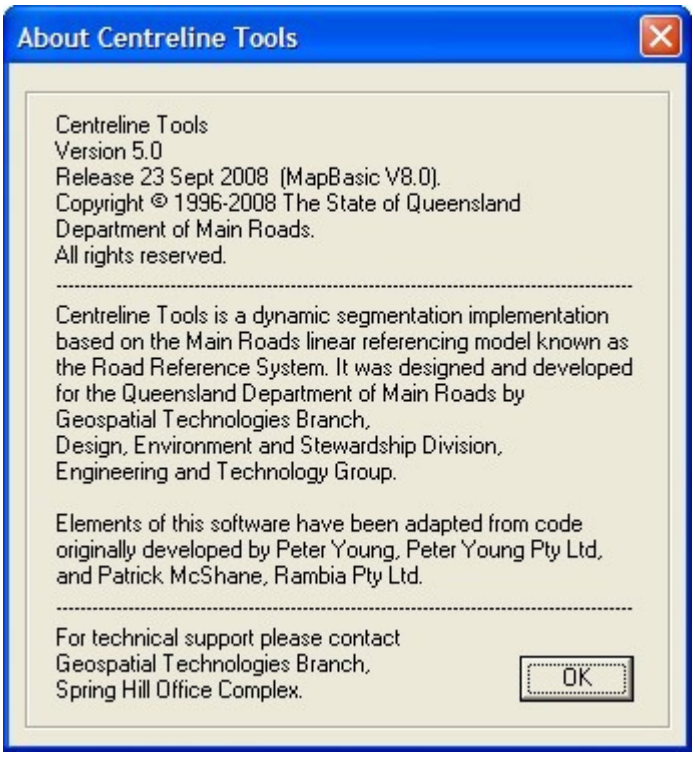


Figure 5.18 – The 'About Centreline Tools' Dialog

5.11 Exit

The ‘Exit’ tool button (see Figure 5.1) closes the system data table *CT_Data* and terminates the program, removing the Centreline Tools toolbar from the MapInfo environment.

CHAPTER 6
CONCLUSIONS

6.1 Introduction

This project delivered an upgraded version of Centreline Tools, a utility used by Main Roads to translate between linear and spatial forms of data.

A system upgrade, particularly one in which the existing look and feel were to be retained, necessarily involved some degree of code reuse. In this project much of the existing code used to manage the toolbar and interface dialogs was reused with varying degrees of modification.

The design of the upgrade was an amalgamation of new and existing ideas. Some of the design elements of the existing version, including the representation of segment direction and the use of an expanding search area were retained but implemented in a different way. However, the methods used to implement dynamic segmentation processes were developed from original ideas.

The organisation and structure of the upgrade represented a significant departure from the existing version and contributed to an expected simplification in program maintenance. Initially, the expanded definition file centralised program information and simplified referencing. Additional advantages ensued.

The module restructure in combination with the introduction of the SegmentType data type and segment filtering, ensured that only required segments were processed. The

defined type also enabled selected segments to be passed between procedures, significantly reducing the need for the regular selections from the *sc_roads* table seen throughout the existing version.

Expanding the use of procedure parameters in the upgrade also helped to minimise what was considered to be an unnecessary dependence on the use of global variables in the existing version.

These changes dramatically improved the logic and flow of the program making it much easier to follow and hence, maintain.

Anecdotal accounts suggest that the upgrade is a definite improvement over the existing version and that it is more intuitive to use. Positive evidentiary reports have also been received.

The test group in Cairns advised that the new version was used to geocode their annual RIP (Roads Implementation Program) in preparation for mapping. A substantial number of tables were processed with no problems (Angus, B 2008, pers. comm., 29 Aug).

A report from Townsville indicated that the Centreline Tools upgrade is working very well (Appleman, A 2008, pers. comm., 10 Sept).

Mackay also indicated that the upgrade is performing well (Clague, S 2008, pers. comm., 17 Sept).

6.2 Benefits

It is a little too early in the life cycle of the upgrade to claim that all the expected benefits have been realised. However, to date feedback has been very positive. In addition, there have been no reports of unexplained program behaviours or crashes.

While an exhaustive assessment is not yet possible, the following benefits can tentatively be claimed:

- The issue of producing an incorrect chainage at first usage has been resolved.

- The status reports provided by the table processing operations are useful.
- Selection of linear reference data formats is more flexible.
- The upgraded version will definitely be easier to maintain.
- Early evidence suggests that the upgrade is consistent, stable and reliable.

A more informed assessment of the benefits provided by the Centreline Tools upgrade project will be possible only after the new version has been in production for a reasonable period of time.

6.3 Future Work

A number of improvements could be made to Centreline Tools in the future to provide a more functional and complete product. Some of these might include:

- enabling offsetting of points to the left and right of the road centreline by a specified distance,
- dynamically displaying labels while geocoding,
- accommodating the use of another road centreline product,
- interactive setting of parameters such as the proximity tolerance for point objects in relation to the road centreline,
- providing a 'Help' facility to assist less experienced users, and
- updating the user guide.

In addition, some form of advice is necessary, to point out that results produced by the program are entirely dependent on the accuracy of the road centreline representation of ARMIS and the Road Reference System.

Future development of Centreline Tools will also inevitably be impacted by other work carried out in Main Roads. For example, future projects or processes that modify the structure, content or availability of *sc_roads* would no doubt impact on Centreline Tools.

In addition, a Road Information Integrator (RII) is currently being developed to perform translations between linear references and spatial coordinates at the system level between ARMIS sub-systems (Mayocchi, W 2008, pers. comm., 4 Feb). While support

for MapInfo is currently not in scope for this project, a service providing spatial/linear translations into the desktop MapInfo environment is indeed likely in the longer term. How this service might be utilised is currently not know, but it could impact significantly on Centreline Tools in the future. Some form of interface would still be necessary, however, through which to request and apply the RII service in a MapInfo environment.

Any proposal for further work on Centreline Tools will also be subject to the usual departmental assessment of need and priority to secure funding.

6.4 Conclusions

All of the specific objectives outlined for this project were achieved, including the additional functionality that was identified as likely to be completed only if time permitted.

Some unexpected problems were encountered during the development phase of the project. Several MapBasic functions did not perform as claimed in certain circumstances and were not sufficiently reliable to be used. This was disappointing and resulted in some compromises in system development.

The use of the iterative approach to the SDLC methodology proved to be very effective in the context of this project. This approach enabled significant periods of operational testing of the logic and procedures at the core of the upgrade. It also allowed useful feedback to be evaluated and integrated into subsequent development releases.

While it is still very early in the life cycle of the upgrade, the majority of the expected benefits from the project have already been realised, and the life of the utility has been extended.

The Centreline Tools Upgrade delivered by this project was released into production on 23rd September 2008.

REFERENCES

Apan, Armando 2002, *GIS2403/E4028 Land management systems study book*, Distance Education Centre, USQ, Toowoomba.

Clarke, KC 2003, *Getting started with geographic information systems* 4th edn, Prentice Hall, New Jersy.

Davis, B 1996, *GIS a visual approach*, OnWord Press, NM, USA.

Department of Main Roads 1999, *MapInfo GIS user guide version 4.0,* Department of Main Roads.

Department of Main Roads 2004, *MapView training workbook*, Department of Main Roads.

Department of Main Roads 2006, *ARMIS road reference system on-line help,* Department of Main Roads.

Department of Main Roads 2007, *Main Roads annual report 2006-07*, Department of Main Roads.

Department of Main Roads 2008*, Main Roads strategic plan 2008 – 2013*, Department of Main Roads.

ESRI 2001, *Linear referencing and dynamic segmentation in ArcGIS 8.1 An ESRI White Paper May 2001*, Redlands, CA, USA.

ESRI 2002, *Linear referencing in ArcGIS*, ESRI, Redlands, CA, USA.

Intergraph Corporation 2001, *Working with GeoMedia*, Intergraph Corporation, Huntsville, Alabama, USA.

Korte, GB 2001, *The GIS book* 5th edn, OnWord Press, NY, USA.

Kothuri, R, Godfrind, A & Beinat E 2004, *Pro oracle spatial*, Apress.

Longley, PA, Goodchild, MF, Maguire, DJ & Rhind, DW 2005, *Geographical information systems and science* 2nd edn, Wiley, West Sussex, England.

MapInfo Corporation 2005, *MapBasic 8.0 development environment reference guide*, MapInfo Corporation, Troy, New York, USA.

MapInfo Corporation 2005a, *MapBasic development environment version 8.0 user guide,* MapInfo Corporation, Troy, New York, USA.

MapInfo Corporation 2005b, *MapInfo Professional version 8.0 user guide*, MapInfo Corporation, Troy, New York, USA.

Montgomery, GE & Schuch, HC 1993, *GIS data conversion handbook*, GIS World Inc., Fort Collins, Colorado, USA.

Scott, M, Holland, J, O'Leary, TJ & O'Leary, LI 1996, 'Systems analysis and design', ch. 11 in *Information Processing and Management*, 2nd edn, McGraw-Hill Book Company, Sydney, pp. 202-17.

Senn, JA 1989, *Analysis & design of information systems*, 2nd edn, McGraw-Hill Book Company, Singapore.

Thompson, RW, Jeffries, RT & McCluskey, M 2006, *Spatial information strategic plan 2006 – 2011*, Department of Main Roads.

Wood, SJ 2000, *A practitioner's guide to GIS terminology 2000 edn*, Data West Research Agency, WA, USA.

Appendix A – Project Specification

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project
PROJECT SPECIFICATION

FOR:	**Robert William THOMPSON**
TOPIC:	Centreline Tools Upgrade
SUPERVISOR:	Mrs Xiaoye Liu
SPONSORSHIP:	Queensland Department of Main Roads
PROJECT AIM:	To upgrade the MapBasic linear segmentation utility known within Main Roads as Centreline Tools.

PROGRAMME: **(Issue A, 20th March 2008)**

1. Examine the current centreline tools utility to determine the range of operations it performs and develop priorities and a feasible scope definition for the upgrade.
2. Research the implementation of the linear referencing system used by Main Roads to describe the physical structure of the state controlled road network.
3. Research the relationship between Main Roads' linear referencing system and the structure of the digital road centreline which is used as a spine for the linear segmentation process.
4. Evaluate the source code of the current system to identify procedures suitable for modification and reuse in the upgrade.
5. Design and implement new algorithms in MapBasic to translate between locations described by linear references and mappable spatial locations based on coordinates.
6. Develop an upgraded centreline tools utility using an amalgamation of new algorithms and modified existing system procedures.
7. Test and distribute the upgraded utility.
8. Prepare and submit an academic dissertation on the system upgrade process.

As time permits:

9. Develop additional functionality for the upgraded utility that was originally out of scope.
10. Include a function to report on processing anomalies.

AGREED _____ (student) Date: 27/03/2008

_____ (supervisor) Date: 03/04/2008

Co-examiner: _____

Appendix A – Project Specification

Appendix B – Centreline Tools Definition File

```
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'@@@@@                                                                      @@@@@
'@@@@@   @@@@@@@@    @@@@@@@@   @@@@@@@@         @@@@@@@    @@   @@@@@@@    @@@@@
'@@@@@   @@     @@  @@  @@  @@  @@     @@       @@      @@  @@  @@          @@@@@
'@@@@@   @@     @@  @@  @@  @@  @@     @@       @@          @@  @@          @@@@@
'@@@@@   @@     @@  @@  @@  @@  @@@@@@@@        @@          @@   @@@@@@@    @@@@@
'@@@@@   @@     @@  @@  @@  @@  @@   @@         @@     @@@  @@         @@   @@@@@
'@@@@@   @@     @@  @@  @@  @@  @@    @@        @@      @@  @@         @@   @@@@@
'@@@@@   @@@@@@@@   @@  @@  @@  @@     @@        @@@@@@@    @@   @@@@@@@    @@@@@
'@@@@@                                                                      @@@@@
'@@@@@                    MainRoads MapInfo GIS Software                    @@@@@
'@@@@@                                                                      @@@@@
'@@@@@                           Centreline Tools                           @@@@@
'@@@@@                                                                      @@@@@
'@@@@@   Designed and developed by                                          @@@@@
'@@@@@   Geospatial Technologies Branch,                                    @@@@@
'@@@@@   Planning Design & Operations Division,                             @@@@@
'@@@@@   Engineering & Technology Group                                     @@@@@
'@@@@@   Spring Hill Office Complex                                         @@@@@
'@@@@@   Queensland Department of Main Roads.                               @@@@@
'@@@@@                                                                      @@@@@
'@@@@@   Elements of this software have been adapted from code              @@@@@
'@@@@@   originally developed by Patrick McShane and Peter Young,           @@@@@
'@@@@@   South Coast Hinterland District, Nerang.                           @@@@@
'@@@@@                                                                      @@@@@
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'@@@@@   Notes:                                                             @@@@@
'@@@@@   1. Copyright © The State of Queensland,                            @@@@@
'@@@@@      Queensland Department of Main Roads.                            @@@@@
'@@@@@   2. This software is authorised for internal Main Roads use only.   @@@@@
'@@@@@   3. Unauthorised modifications will void support arrangements.      @@@@@
'@@@@@                                                                      @@@@@
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
'////////////////////////////////////////////////////////////////////////////////
'////////////////////////////////////////////////////////////////////////////////
' Centreline Tools Utility Definition File
' Utility definitions and declarations are all recorded in this definition file
' to centralise system information for easier reference and to simplify
' maintenance.
' This also avoids the use-before-declare issues that can occur because of the
' order of locally declared procedures.
' RWT June 2008.
'////////////////////////////////////////////////////////////////////////////////



'--------------------------------------------------------------------------------
' Data type definition representing a road centreline segment object and the
' attributes necessary to translate between coordinates and linear references.
' This is used to make procedures involving centreline manipulations more
' independent. Selections on the centreline tables for a particular operation
' are then necessary in one procedure only. Selected segments are then carried
' as arrays of the segment type.
' This type can therefore potentially be used to pass segment selections
' from any road centreline table containing Road Reference System attributes.
'--------------------------------------------------------------------------------
TYPE SegmentType
   SegObj        As Object                   '- segment object
   SegRoad       As String                   '- segment road identifier
   SegCway       As String                   '- segment carriageway code
   SegStart      As Float                    '- segment start chainage in m
   SegLength     As Float                    '- segment driven length in m
   SegScale      As Float                    '- segment scalefactor
END TYPE



'--------------------------------------------------------------------------------
' Constant definitions
'--------------------------------------------------------------------------------
DEFINE CT_VERSION             "5.0"          '- version identifier
DEFINE CT_RELEASE_DATE        "23 Sept 2008" '- version release date

DEFINE CT_DEBUG_ON            FALSE  'TRUE   '- set T in development mode for info
DEFINE CT_DEBUG_TAB           "Debug"        '- debugging table name

DEFINE CT_PRECISION           50.0           '- point to road fail distance in m
```

```
DEFINE CT_SC_ROADS             "sc_roads"     '- road centreline table name

DEFINE CT_DATA_TAB             "CT_Data"      '- temp table for objects/labels
DEFINE CT_SECT_DEF             "Section"      '- temp table for section definition
DEFINE CT_SECT_ELEMENTS        "Elements"     '- temp table for section elements

DEFINE CT_DIR_WITH_GAZ         1              '- line direction with gazettal
DEFINE CT_DIR_AGAINST_GAZ      2              '- line direction against gazettal
DEFINE CT_DIR_UNKNOWN          3              '- line direction unspecified

DEFINE CT_CWAY_BOTH            "1"            '- cway code in both directions
DEFINE CT_CWAY_WITH_GAZ        "24ABCDEKLMXZ" '- cway codes in gazettal direction
DEFINE CT_CWAY_AGAINST_GAZ     "35NOPQRSTUY"  '- cway codes against gazettal

DEFINE CT_UNITS_M              1              '- represents unit of measure of metres
DEFINE CT_UNITS_KM             2              '- represents unit of measure of km


'--------------------------------------------------------------------------------
' Global variable declarations
'--------------------------------------------------------------------------------
GLOBAL gSysApplicationDir$     As String      '- SAS system directories
GLOBAL gSysWorkspaceDir$       As String      '-  "
GLOBAL gSysDataDir$            As String      '-  "
GLOBAL gSysReadWrite$          As String      '-  "


'================================================================================
' CT_Main.MB procedure declarations
'================================================================================
DECLARE      SUB AboutCT
DECLARE      SUB DisplayClTab(ByVal sInTable As String)

DECLARE      SUB DisplayLabel(ByVal oInputObj As Object,
                              ByVal sClTable  As String,
                              ByVal sInputTable As String,
                              ByVal sRoadId As String,
                              ByVal sCarrWay As String,
                              ByVal sStartChainage As String,
                              ByVal sEndChainage As String,
                              ByVal sLength As String,
                              ByVal sLabelExpr As String)

DECLARE      SUB ExitCT
DECLARE      SUB LoadCTButtonPad
DECLARE      SUB Main
DECLARE FUNCTION OpenClTab(ByVal sInTable As String) As Logical
DECLARE      SUB RemoveCTLabels
DECLARE      SUB SaveCTLabels
DECLARE      SUB SetSectionButtons
DECLARE      SUB ResetToolButtons


'================================================================================
' CT_CoordsToRef.MB procedure declarations
'================================================================================
DECLARE FUNCTION CaptureClickPoint(oMouseClickPoint As Object) As Logical

DECLARE FUNCTION ChainageAtNode(ByVal iNodeNumber As Integer,
                                ByVal oSegment As Object,
                                ByVal sCarrway As String,
                                ByVal fStartChainage,
                                ByVal fDrivenLength,
                                ByVal fScalefactor As Float) As Float

DECLARE      SUB PointToReference


'================================================================================
' CT_Debug.MB procedure declarations
'================================================================================
DECLARE      SUB MakeDebugTable
```

```
'================================================================================
' CT_Dialogs.MB procedure declarations
'================================================================================
DECLARE      SUB BrowseSelectInTable

DECLARE FUNCTION CopyInputTable(ByVal sInputTable As String,
                                sNewTable As String) As Logical

DECLARE      SUB ResetSelectColumns
DECLARE      SUB ResetSelectInTable

DECLARE FUNCTION SelectColumns(ByVal sLinearFile As String,
                               iLinearFormat As SmallInt,
                               asColumnNames(0) As String,
                               iSectionUnits,
                               iLengthUnits As SmallInt) As Logical

DECLARE FUNCTION SelectInputTable(ByVal iDialogType As SmallInt,
                                  sInputFile As String) As Logical

DECLARE      SUB SetSelectColumns
DECLARE      SUB SetSelectInTable
DECLARE      SUB VerifySelectColumns



'================================================================================
' CT_DynSeg.MB procedure declarations
'================================================================================
DECLARE      SUB ProcessPointTable
DECLARE      SUB ProcessReferenceTable



'================================================================================
' CT_FindNodes.MB procedure declarations
'================================================================================
DECLARE FUNCTION ConstructPortion(ByVal fSectionStart, fSectionEnd As Float,
                                  aSectionSeg As SegmentType,
                                  fSegPortionStart,
                                  fSegPortionEnd As Float) As Object

DECLARE FUNCTION FindNodeAtChainage(ByVal fChainage As Float,
                                    oLinearObj As Object,
                                    ByVal sSegCway As String,
                                    ByVal fSegRefoffset As Float,
                                    ByVal fSegReflen As Float,
                                    ByVal fScalefactor As Float) As Integer

DECLARE FUNCTION FindNodeAtPoint(ByVal oPointArea As Object,
                                 oLinearObj As Object) As Integer

DECLARE FUNCTION SegmentDirection(ByVal sCarrwayCode As String) As SmallInt
DECLARE      SUB VerifyPolyline(oObject As Object)



'================================================================================
' CT_RefToCoords.MB procedure declarations
'================================================================================
DECLARE FUNCTION EnterLinearReference(sRoadIn As String,
                                      fChainageIn As Float) As Logical

DECLARE      SUB ReferenceToPoint

DECLARE FUNCTION SelectReferenceCway(aSegmentsFound() As SegmentType,
                                     sSelectOption As String) As Logical

DECLARE      SUB VerifyLinearReference
DECLARE      SUB VerifyRoad



'================================================================================
' CT_SectionByCoords.MB procedure declarations
'================================================================================
DECLARE      SUB CaptureSectionCoord
DECLARE      SUB SaveSectionCoords
DECLARE      SUB SectionByCoords
DECLARE FUNCTION VerifyCoordSection() As Logical
```

```
'================================================================================
' CT_SectionByRefs.MB procedure declarations
'================================================================================
DECLARE FUNCTION EnterReferenceSection(sSectionRoad As String,
                                       fSectionStart,
                                       fSectionEnd As Float) As Logical

DECLARE      SUB ExtractSection(ByVal sSectionRoad As String,
                                ByVal fSectionStart, fSectionEnd As Float)

DECLARE      SUB MakeElementTable
DECLARE      SUB ProcessSectionElements
DECLARE      SUB ResetBuffer
DECLARE      SUB SectionByReferences

DECLARE FUNCTION SelectSectionProcess(bIndividual,
                                      bGrouped,
                                      bCombined,
                                      bAddBuffer As Logical,
                                      fNewRadius As Float) As Logical

DECLARE      SUB VerifyReferenceSection
DECLARE      SUB WriteCombinedCways
DECLARE      SUB WriteCombinedSection
DECLARE      SUB WriteIndElements
DECLARE      SUB WriteSectionBuffer(ByVal fBufferRadius As Float)

DECLARE      SUB WriteSectionElement(ByVal oElementObj As Object,
                                     ByVal sElementRoad, sElementCway As String,
                                     ByVal fElementStart, fElementEnd As Float)



'================================================================================
' CT_Select.MB procedure declarations
'================================================================================
DECLARE FUNCTION FindPointSegments(ByVal oStartPoint As Object,
                                   oSearchArea As Object,
                                   aPointSegs() As SegmentType) As Logical

DECLARE FUNCTION FindReferenceCways(ByVal sRoadId As String,
                                    ByVal fChainage As Float,
                                    aCarriageways() As String) As Logical

DECLARE FUNCTION FindReferenceSegments(ByVal sRoadId As String,
                                       ByVal sCarriageway As String,
                                       ByVal fChainage As Float,
                                       aReferenceSegs() As SegmentType) As Logical

DECLARE FUNCTION FindSectionCways(ByVal sRoadId As String,
                                  ByVal fSectionStart, fSectionEnd As Float,
                                  aCarriageways() As String) As Logical

DECLARE FUNCTION FindSectionSegments(ByVal sRoadId As String,
                                     ByVal sCarriageway As String,
                                     ByVal fSectionStart, fSectionEnd As Float,
                                     aSectionSegs() As SegmentType) As Logical

DECLARE FUNCTION RoadExists(ByVal sRoadId As String) As Logical

DECLARE      SUB RoadExtents(ByVal sRoadId As String,
                             fMinExtent, fMaxExtent As Float)
'--------------------------------------------------------------------------------
' End of definition file
'--------------------------------------------------------------------------------
```

Appendix B – Centreline Tools Definition File

Appendix C – Procedure Structure Charts

C.A – Sub-procedure Main Structure Chart

C.B – Sub-procedure PointToReference Structure Chart

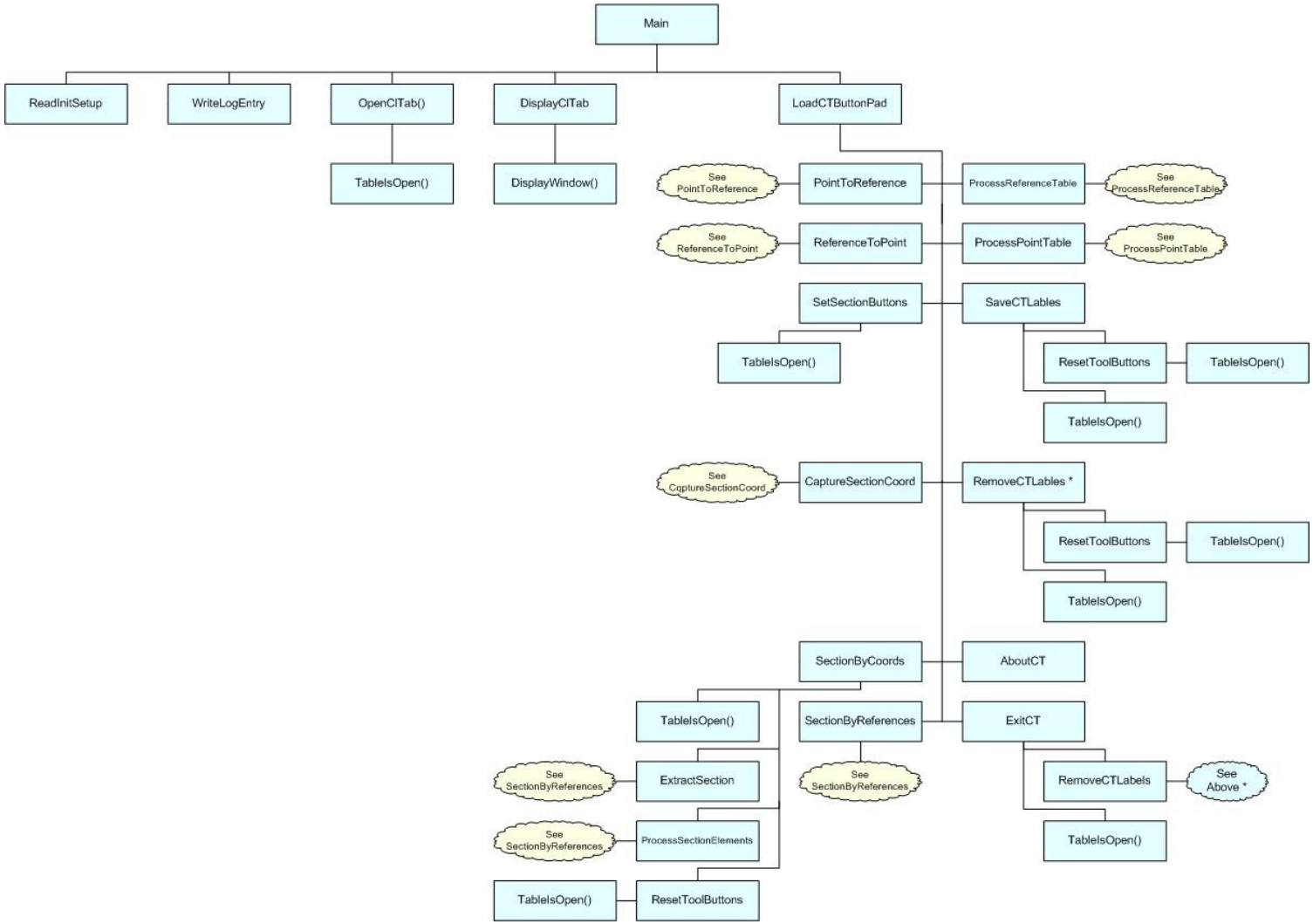
C.C – Sub-procedure ReferenceToPoint Structure Chart

C.D – Sub-procedure ProcessReferenceTable Structure Chart

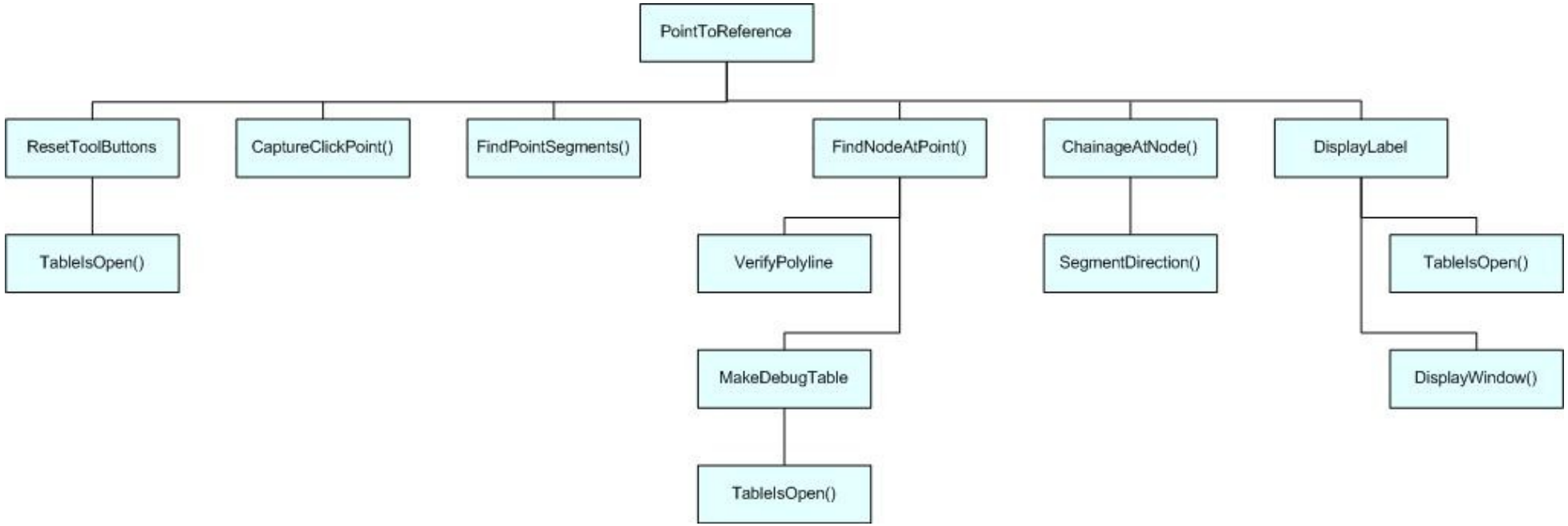
C.E – Sub-procedure ProcessPointTable Structure Chart

C.F – Sub-procedure CaptureSectionCoord Structure Chart

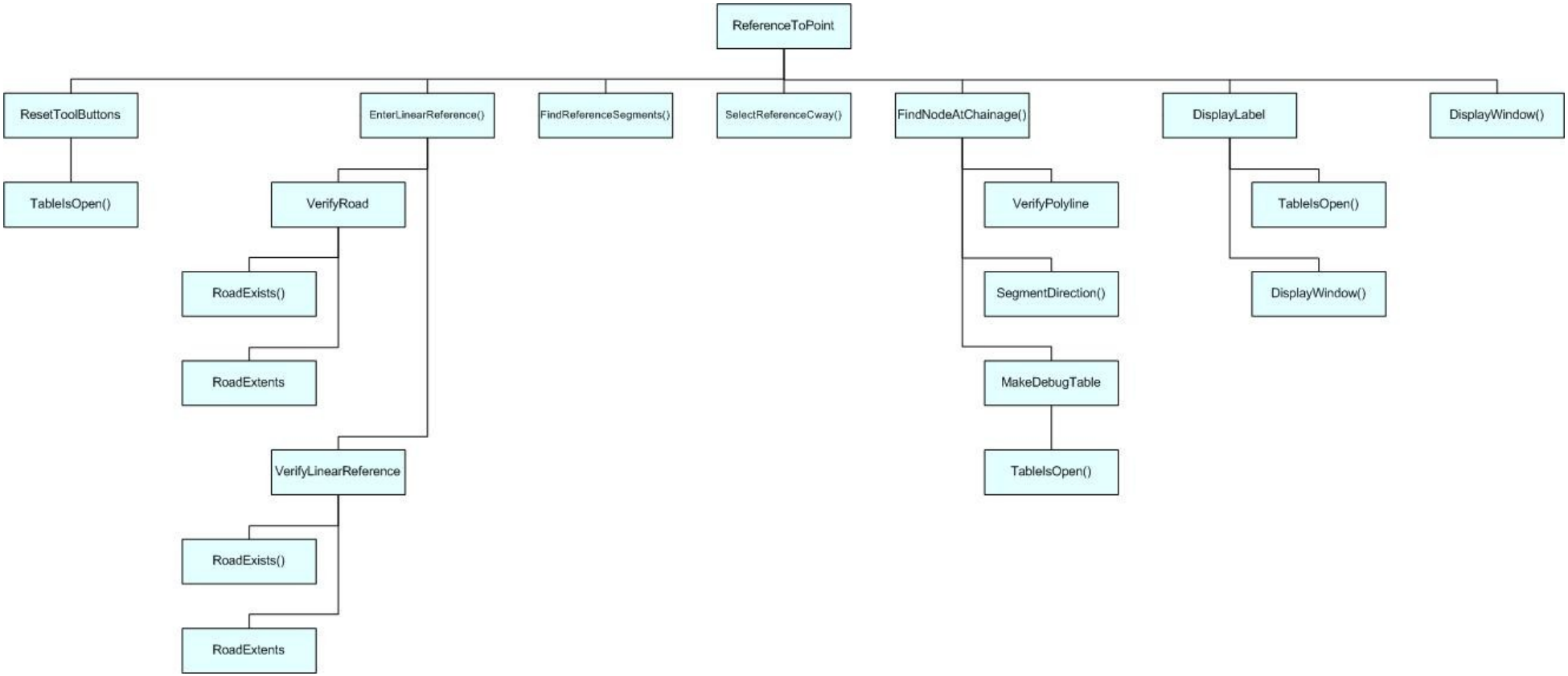
C.G – Sub-procedure SectionByReferences Structure Chart



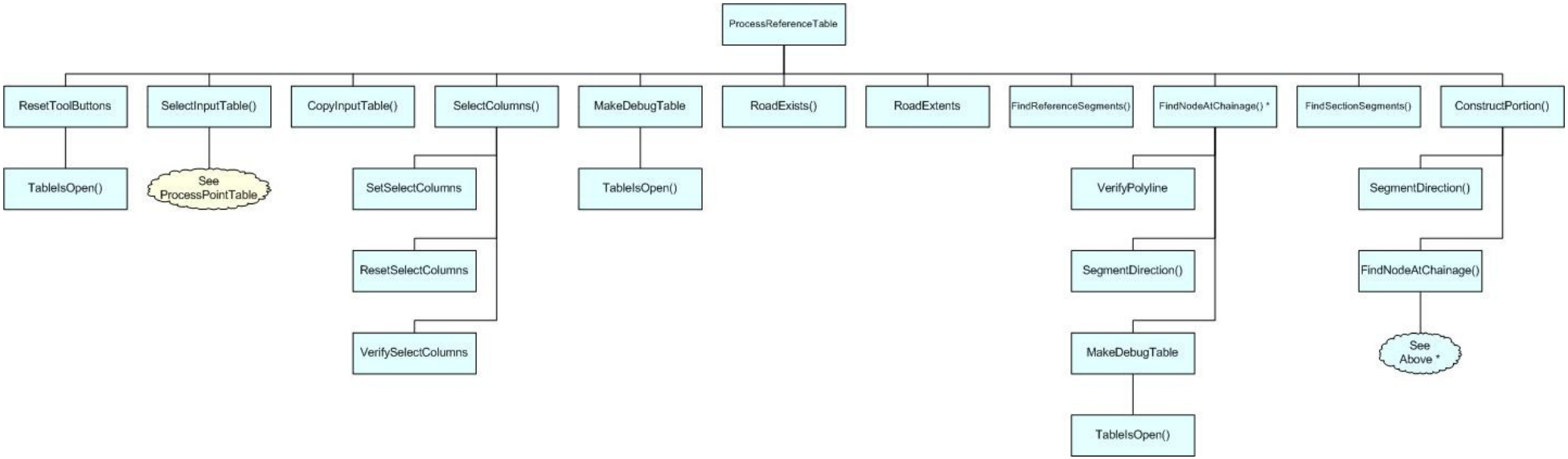
Appendix C.A – Sub-procedure Main Structure Chart



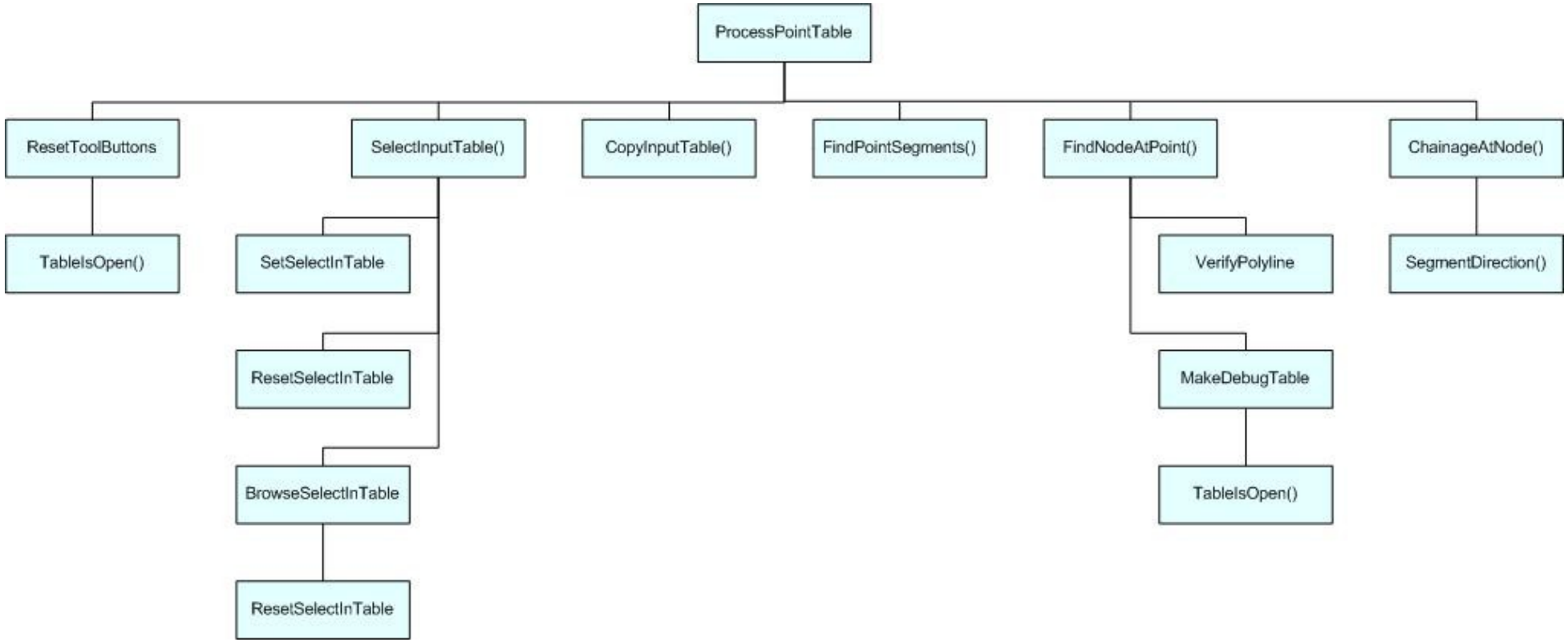
Appendix C.B – Sub-procedure PointToReference Structure Chart



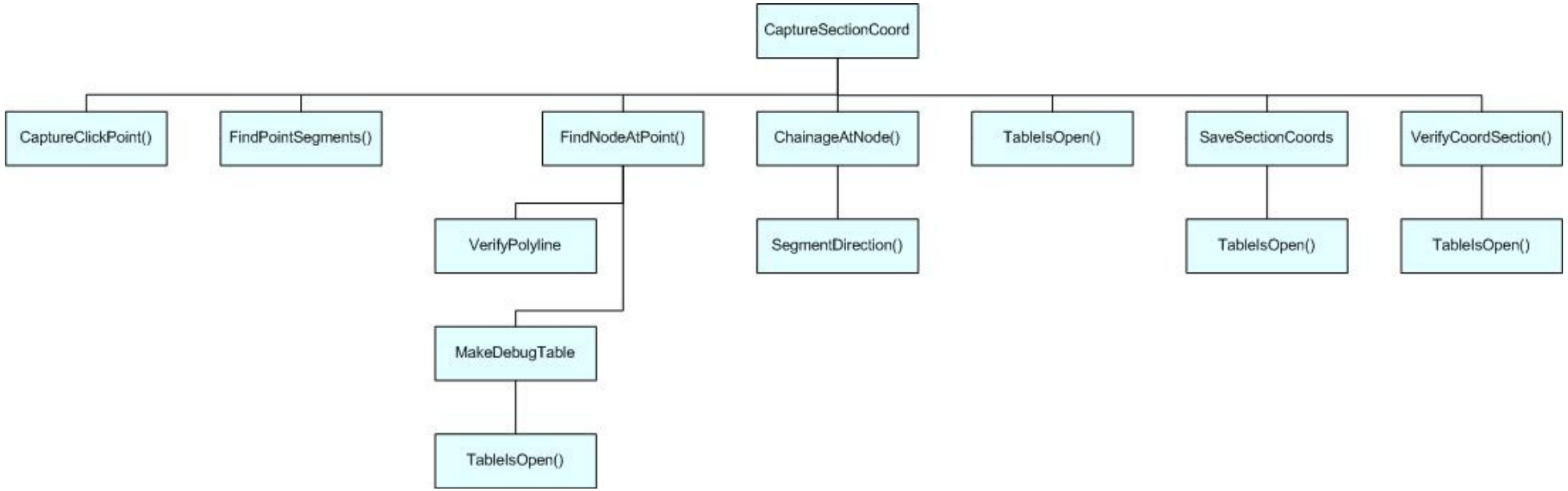
Appendix C.C – Sub-procedure ReferenceToPoint Structure Chart



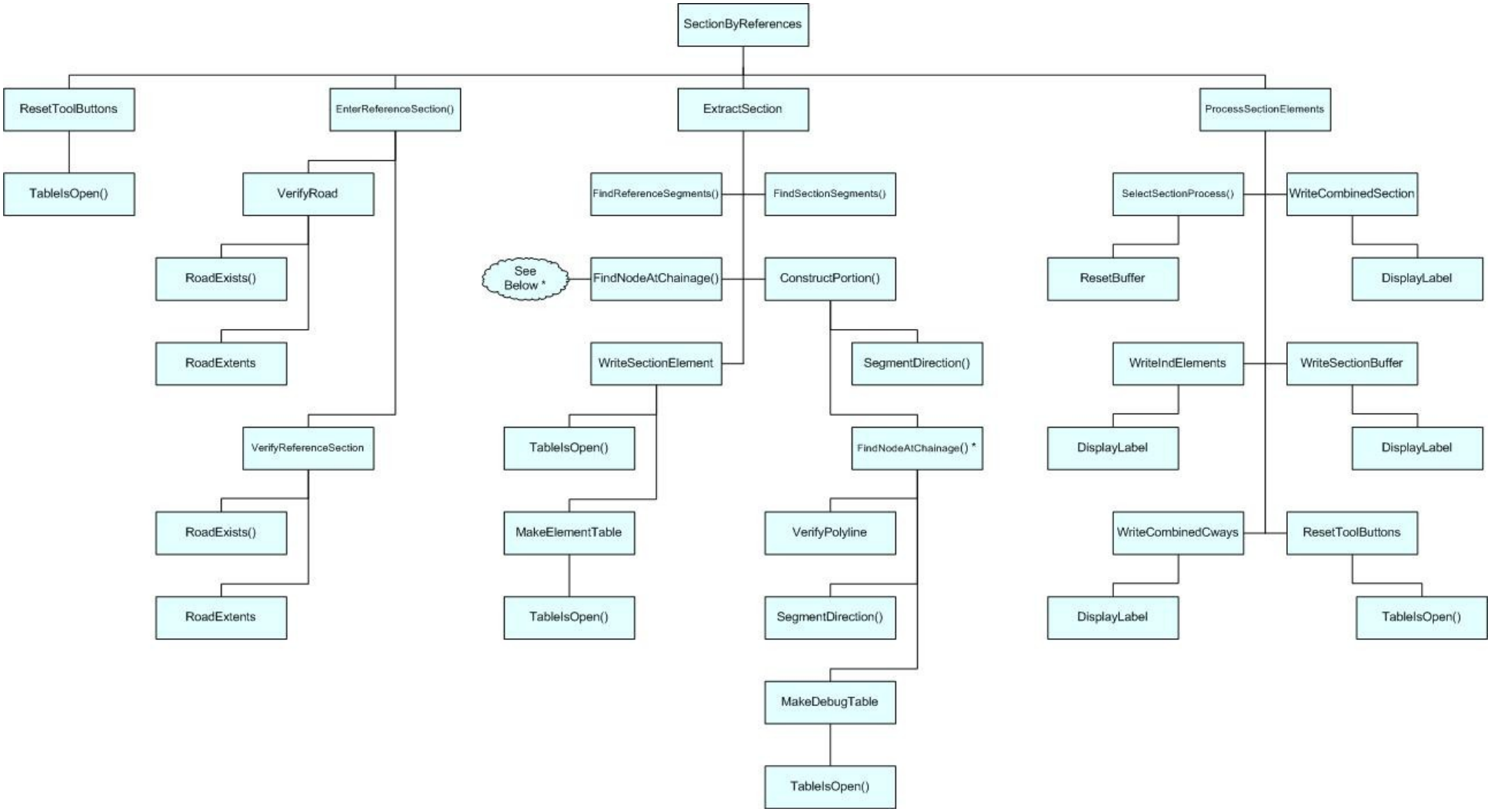
Appendix C.D – Sub-procedure ProcessReferenceTable Structure Chart



Appendix C.E – Sub-procedure ProcessPointTable Structure Chart



Appendix C.F – Sub-procedure CaptureSectionCoord Structure Chart



Appendix C.G – Sub-procedure SectionByReferences Structure Chart

Appendix D – Function FindReferenceSegments()

```
'================================================================================
' Finds the road centreline segments in which the named linear reference
' description occurs.
' Selects the road centreline segments for the specified road identifier which
' occur at the specified carriageway (may be one carriageway code or "All" or
' "Through" 1, 2 or 3), have a start chainage <= the specified chainage, and
' an end chainage >= the specified chainage (> does not select end segments).
' Loads the selected segments into a segment type array.
' Loads only one segment for the given road identifier per carriageway
' Returns T if segments are selected, F otherwise.
'================================================================================
FUNCTION FindReferenceSegments(ByVal sRoadId As String,
                               ByVal sCarriageway As String,
                               ByVal fChainage As Float,
                               aReferenceSegs() As SegmentType) As Logical

   DIM i,
       iSegs  As Integer


   OnError GoTo ErrorHandle

   '--- initialise
   FindReferenceSegments = FALSE

   '--- select the centreline segments in which the named linear reference occurs
   '--- ie the segments of the specified road & carriageway for which the
   '--- specified chainage >= the chainage at the start of the segment, and the
   '--- specified chainage <= the chainage at the end of the segment
   '--- (<= is necessary to include end-of-road segments in the selection)
   '--- order the selected segments by carriageway and start chainage (refoffset)

   If sCarriageway = "All" Then

      Select * From CT_SC_ROADS
      Where STREET = sRoadId AND
            REFOFFSET <= fChainage AND
            REFOFFSET + REFLEN >= fChainage
      Into tblReferenceSegs Noselect
      Order By CARRWAY, REFOFFSET

   ElseIf sCarriageway = "Through" Then

      Select * From CT_SC_ROADS
      Where STREET = sRoadId AND
            (CARRWAY = "1" OR
             CARRWAY = "2" OR
             CARRWAY = "3") AND
            REFOFFSET <= fChainage AND
            REFOFFSET + REFLEN >= fChainage
      Into tblReferenceSegs Noselect
      Order By CARRWAY, REFOFFSET

   Else

      Select * From CT_SC_ROADS
      Where STREET = sRoadId AND
            CARRWAY = sCarriageway AND
            REFOFFSET <= fChainage AND
            REFOFFSET + REFLEN >= fChainage
      Into tblReferenceSegs Noselect
      Order By CARRWAY, REFOFFSET

   End If


   '--- if the specified chainage occurs at a reference point
   '--- ie the common point between two segments
   '--- two segments on the one carriageway will be selected
   '--- so exclude the second segment on the same carriageway
   '--- to avoid creating duplicate points with labels
   '--- load the selected segments into an array
   i = 1
   iSegs = TableInfo(tblReferenceSegs, TAB_INFO_NROWS)

   If iSegs > 0 Then
      '--- centreline segments were selected

      '--- load the first segment into the array
      Fetch First From tblReferenceSegs
```

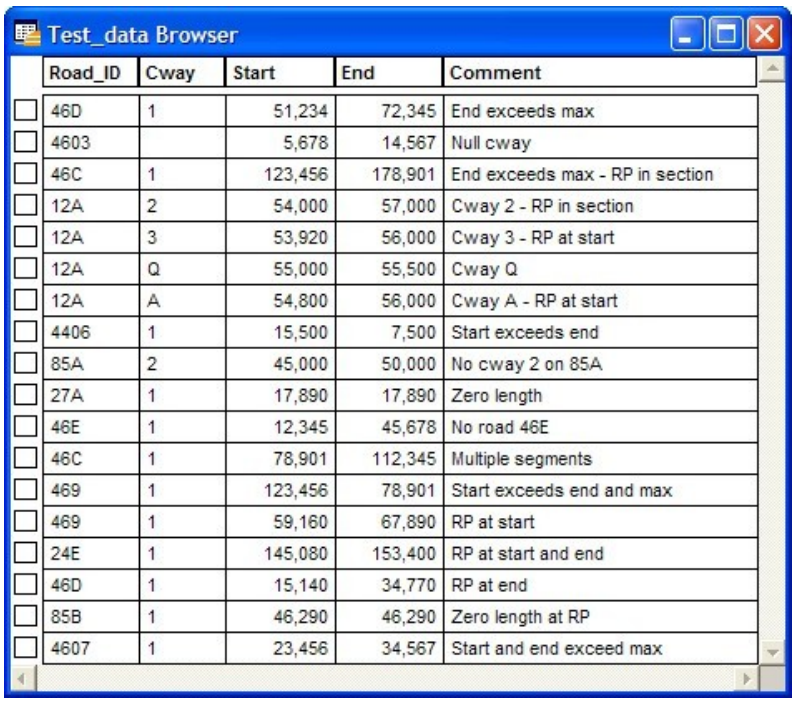
```
      ReDim aReferenceSegs(i)
      aReferenceSegs(i).SegObj    = tblReferenceSegs.Obj
      aReferenceSegs(i).SegRoad   = tblReferenceSegs.STREET
      aReferenceSegs(i).SegCway   = tblReferenceSegs.CARRWAY
      aReferenceSegs(i).SegStart  = tblReferenceSegs.REFOFFSET
      aReferenceSegs(i).SegLength = tblReferenceSegs.REFLEN
      aReferenceSegs(i).SegScale  = tblReferenceSegs.SCALEFACTOR

      '--- avoid divide by zero errors
      If aReferenceSegs(i).SegScale = 0 Then
         aReferenceSegs(i).SegScale = 1
      End If

      '--- loop through the remaining segments and
      '--- load only unique carriageways

      Fetch Next From tblReferenceSegs

      Do While NOT EOT(tblReferenceSegs)

         If tblReferenceSegs.CARRWAY <> aReferenceSegs(i).SegCway Then
            '--- this carriageway is not the same as the previous one
            '--- store this one

            i = i + 1

            ReDim aReferenceSegs(i)
            aReferenceSegs(i).SegObj    = tblReferenceSegs.Obj
            aReferenceSegs(i).SegRoad   = tblReferenceSegs.STREET
            aReferenceSegs(i).SegCway   = tblReferenceSegs.CARRWAY
            aReferenceSegs(i).SegStart  = tblReferenceSegs.REFOFFSET
            aReferenceSegs(i).SegLength = tblReferenceSegs.REFLEN
            aReferenceSegs(i).SegScale  = tblReferenceSegs.SCALEFACTOR

            '--- avoid divide by zero errors
            If aReferenceSegs(i).SegScale = 0 Then
               aReferenceSegs(i).SegScale = 1
            End If
         End If

         Fetch Next From tblReferenceSegs

      Loop

      '--- set return value
      FindReferenceSegments = TRUE

   End If

   Close Table tblReferenceSegs

   GoTo TheEnd
   ErrorHandle:
      Print Error$()
      Resume Next
   TheEnd:

END FUNCTION
```

Appendix D – Function FindReferenceSegments()

Appendix E – Example Test Data

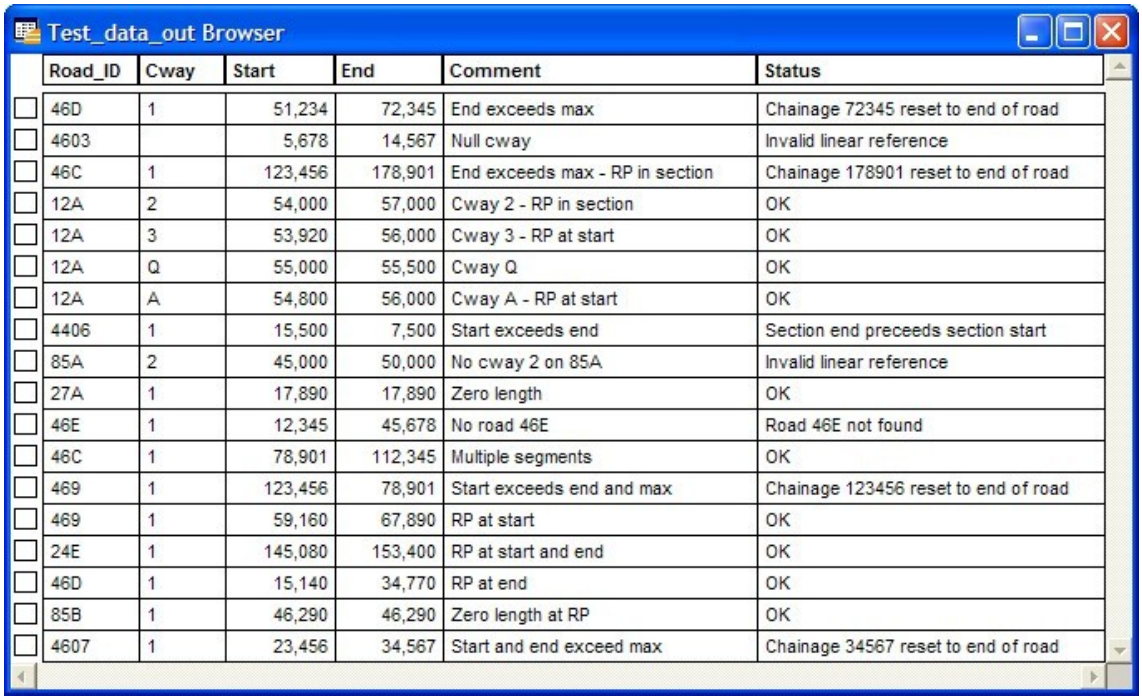
E.A – Test Data Before and After Processing

E.B – Dynamically Segmented Test Data



Test_data Browser

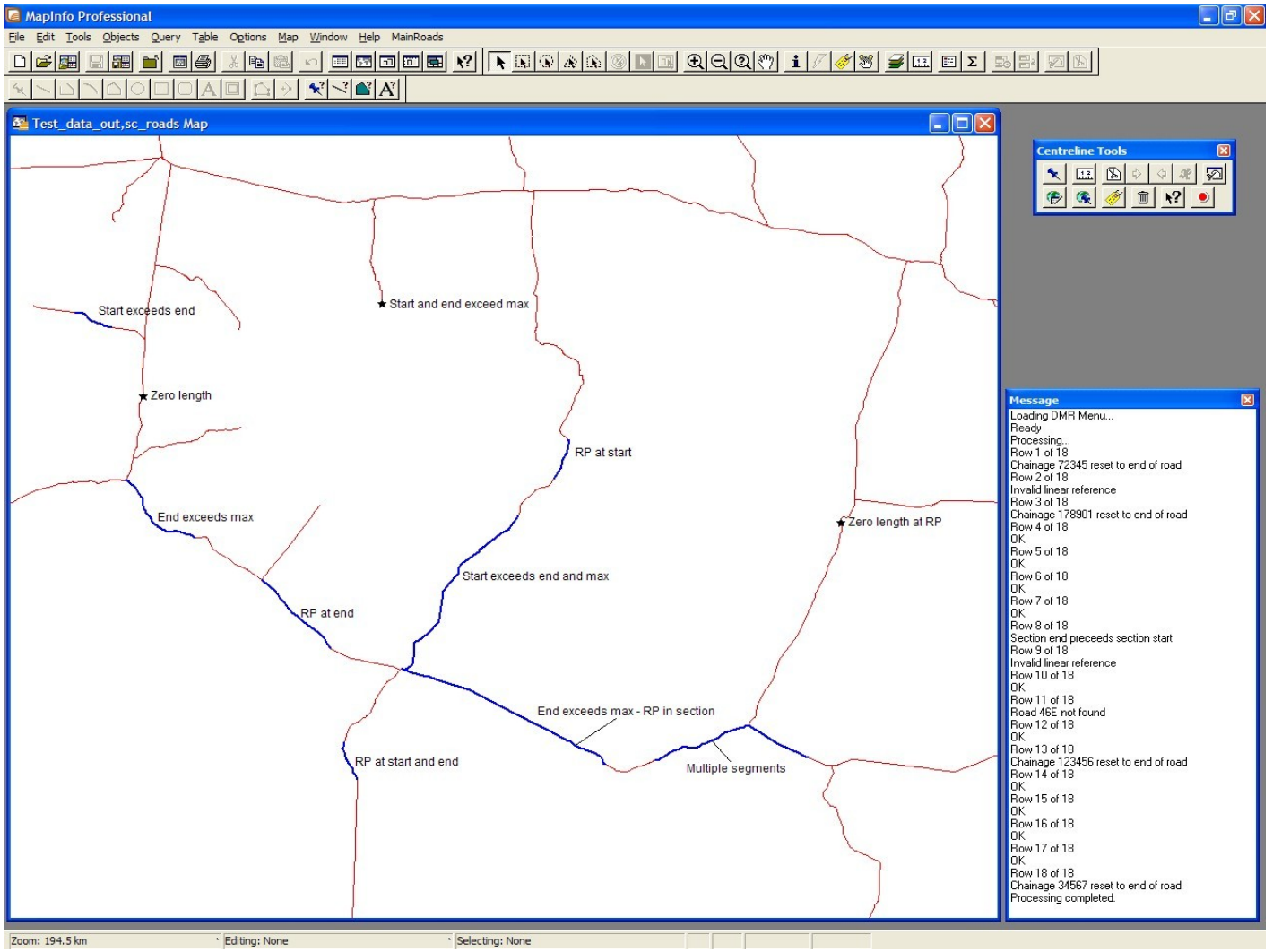
Road_ID	Cway	Start	End	Comment
46D	1	51,234	72,345	End exceeds max
4603		5,678	14,567	Null cway
46C	1	123,456	178,901	End exceeds max - RP in section
12A	2	54,000	57,000	Cway 2 - RP in section
12A	3	53,920	56,000	Cway 3 - RP at start
12A	Q	55,000	55,500	Cway Q
12A	A	54,800	56,000	Cway A - RP at start
4406	1	15,500	7,500	Start exceeds end
85A	2	45,000	50,000	No cway 2 on 85A
27A	1	17,890	17,890	Zero length
46E	1	12,345	45,678	No road 46E
46C	1	78,901	112,345	Multiple segments
469	1	123,456	78,901	Start exceeds end and max
469	1	59,160	67,890	RP at start
24E	1	145,080	153,400	RP at start and end
46D	1	15,140	34,770	RP at end
85B	1	46,290	46,290	Zero length at RP
4607	1	23,456	34,567	Start and end exceed max



Test_data_out Browser

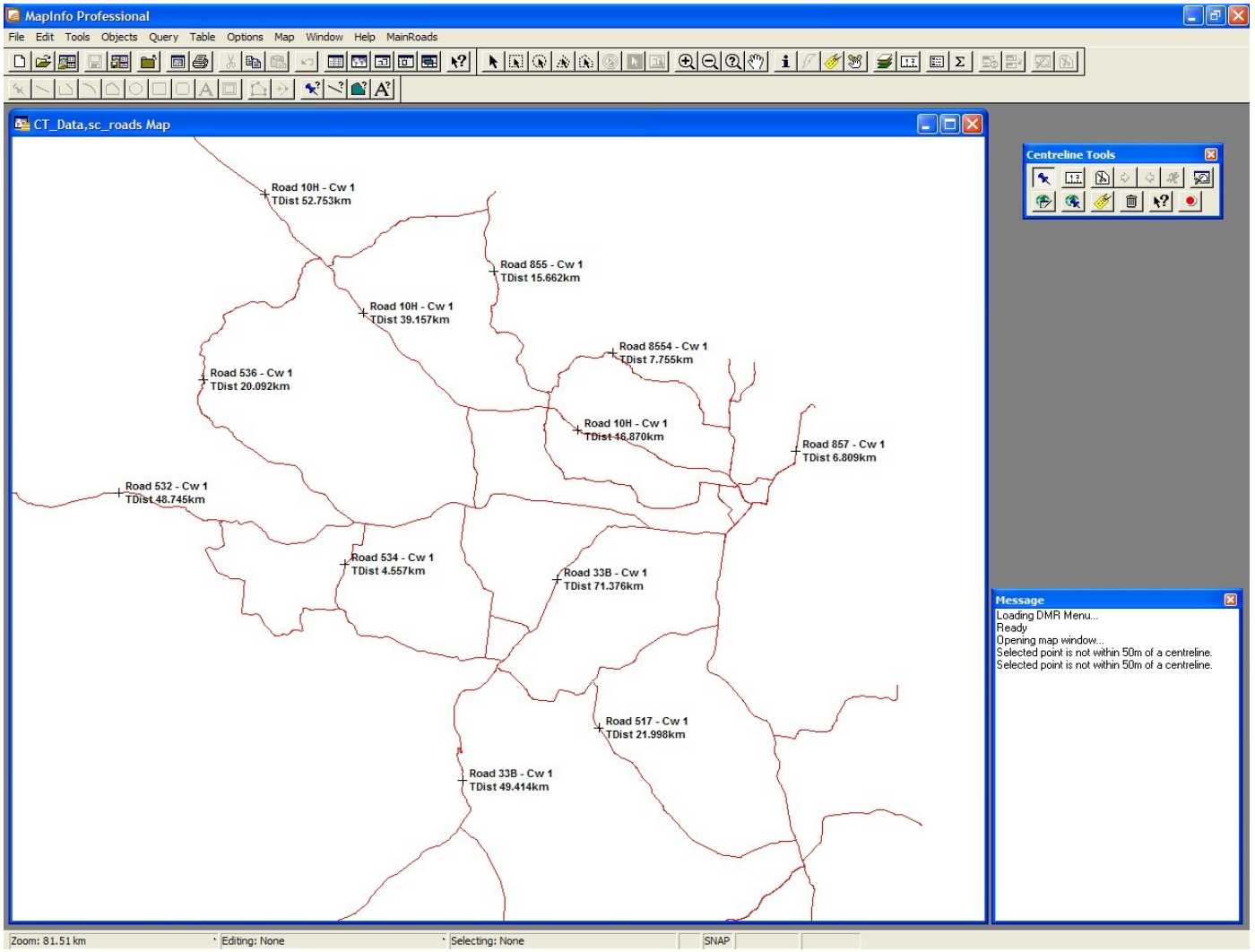
Road_ID	Cway	Start	End	Comment	Status
46D	1	51,234	72,345	End exceeds max	Chainage 72345 reset to end of road
4603		5,678	14,567	Null cway	Invalid linear reference
46C	1	123,456	178,901	End exceeds max - RP in section	Chainage 178901 reset to end of road
12A	2	54,000	57,000	Cway 2 - RP in section	OK
12A	3	53,920	56,000	Cway 3 - RP at start	OK
12A	Q	55,000	55,500	Cway Q	OK
12A	A	54,800	56,000	Cway A - RP at start	OK
4406	1	15,500	7,500	Start exceeds end	Section end preceeds section start
85A	2	45,000	50,000	No cway 2 on 85A	Invalid linear reference
27A	1	17,890	17,890	Zero length	OK
46E	1	12,345	45,678	No road 46E	Road 46E not found
46C	1	78,901	112,345	Multiple segments	OK
469	1	123,456	78,901	Start exceeds end and max	Chainage 123456 reset to end of road
469	1	59,160	67,890	RP at start	OK
24E	1	145,080	153,400	RP at start and end	OK
46D	1	15,140	34,770	RP at end	OK
85B	1	46,290	46,290	Zero length at RP	OK
4607	1	23,456	34,567	Start and end exceed max	Chainage 34567 reset to end of road

Appendix E.A – Test Data Before and After Processing



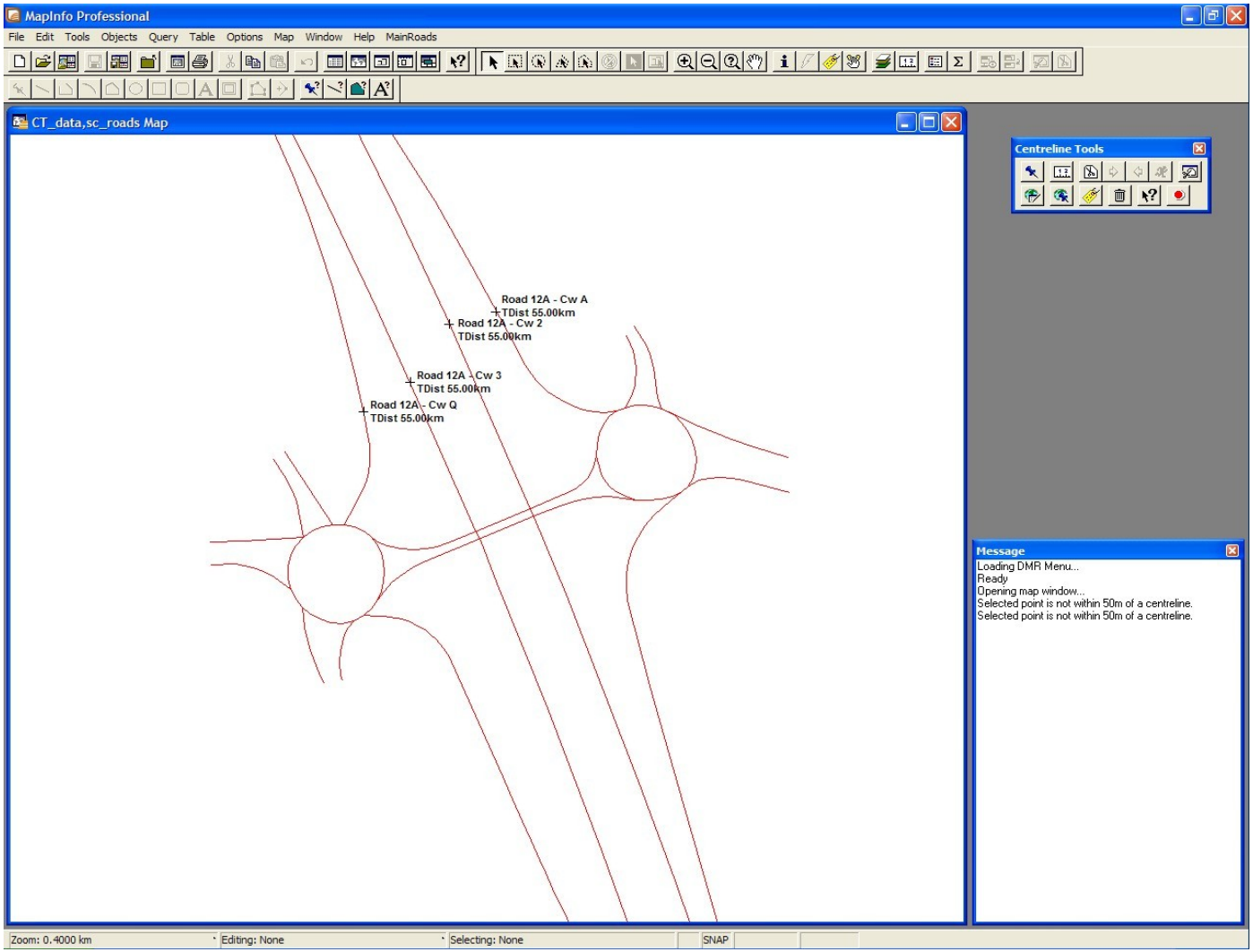
Appendix E.B – Dynamically Segmented Test Data

Appendix F – Linear References at Specified Points



Appendix F – Linear References at Specified Points

Appendix G – Linear References Located

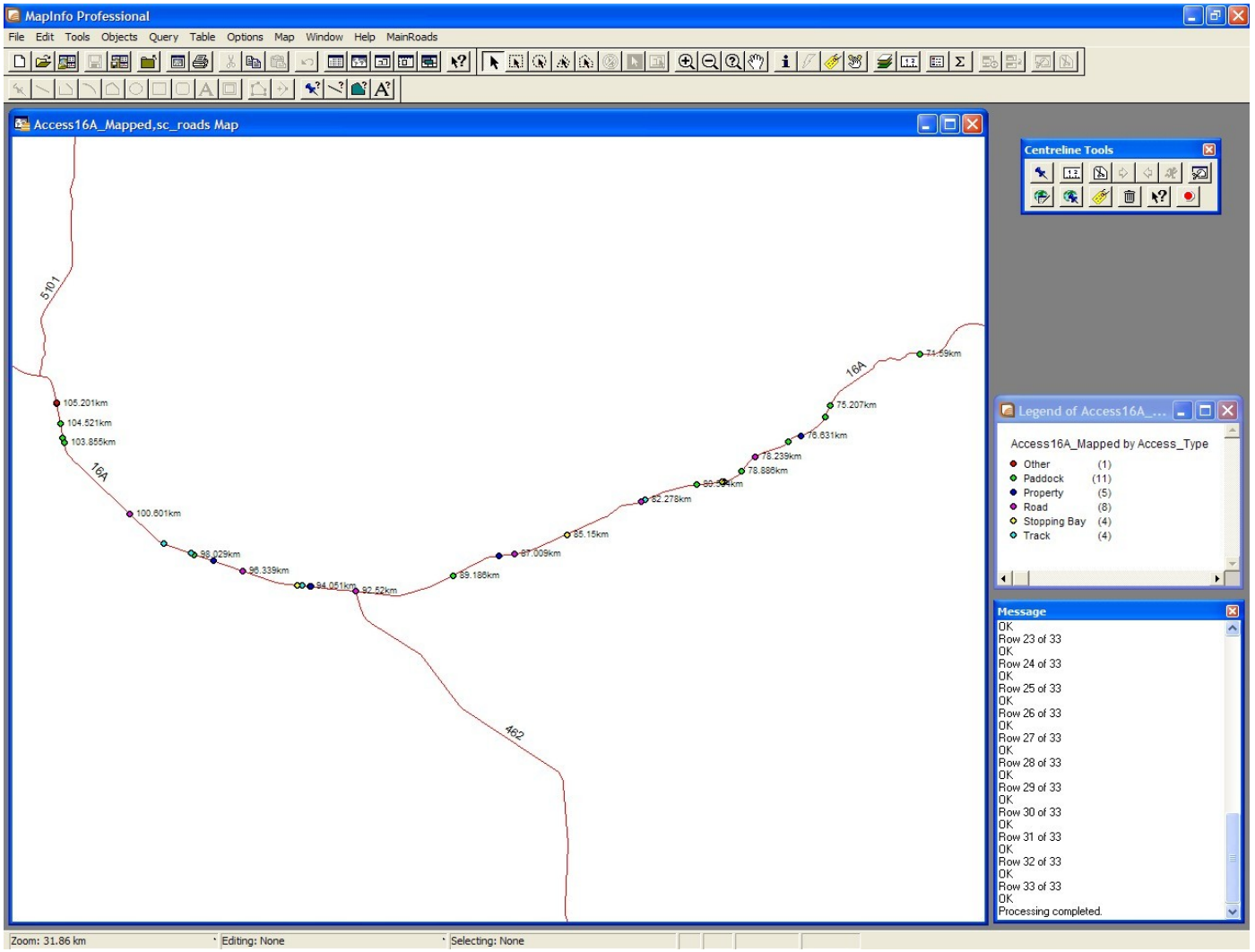


Appendix G – Linear References Located

Appendix H – Dynamic Segmentation

H.A – Dynamically Segmented Linear References

H.B – Attributes Before and After Segmentation



Appendix H.A – Dynamically Segmented Linear References

Access16A Browser

Road_ID	TDist	Side	Access_Type	Comments
16A	71.59	R	Paddock	Rural Address 7116
16A	75.207	L	Paddock	Macintosh Road
16A	75.6	R	Paddock	
16A	76.631	R	Property	Rural Address 7674
16A	77.08	L	Paddock	
16A	78.239	R	Road	Boolburra-Edungalba Road
16A	78.886	L	Paddock	
16A	79.586	R	Stopping Bay	Truck/Car Stopping Bay
16A	79.645	L	Stopping Bay	Truck/Car Stopping Bay
16A	80.504	R	Paddock	
16A	82.278	R	Track	Dirt Track
16A	82.436	L	Road	Anchor Road
16A	85.15	L	Stopping Bay	Truck pull-off area
16A	87.009	L	Road	Prouds Road
16A	87.528	R	Property	
16A	89.186	R	Paddock	
16A	92.52	L	Road	Biloela – Duaringa Road
16A	92.52	R	Road	Biloela-Duaringa Road
16A	94.051	R	Road	Cabarita Road
16A	94.01	L	Property	
16A	94.281	L	Track	Gravel Track
16A	94.468	R	Stopping Bay	Truck pull-off area
16A	96.339	L	Road	Road to Baralaba
16A	97.349	R	Property	
16A	98.029	R	Paddock	
16A	98.147	L	Track	Dirt Track
16A	99.098	L	Track	Dirt/Gravel Track
16A	100.601	L	Road	Laguna Road "Laguna"
16A	103.855	R	Paddock	
16A	104.035	R	Paddock	Rural Address 10404
16A	104.521	R	Paddock	
16A	105.201	R	Property	Monomeath Road. "Duaringa Selling Complex"
16A	105.235	L	Other	Parthenium Weed Washdown Facility

Access16A_Mapped Browser

Road_ID	TDist	Side	Access_Type	Comments	Status
16A	71.59	R	Paddock	Rural Address 7116	OK
16A	75.207	L	Paddock	Macintosh Road	OK
16A	75.6	R	Paddock		OK
16A	76.631	R	Property	Rural Address 7674	OK
16A	77.08	L	Paddock		OK
16A	78.239	R	Road	Boolburra-Edungalba Road	OK
16A	78.886	L	Paddock		OK
16A	79.586	R	Stopping Bay	Truck/Car Stopping Bay	OK
16A	79.645	L	Stopping Bay	Truck/Car Stopping Bay	OK
16A	80.504	R	Paddock		OK
16A	82.278	R	Track	Dirt Track	OK
16A	82.436	L	Road	Anchor Road	OK
16A	85.15	L	Stopping Bay	Truck pull-off area	OK
16A	87.009	L	Road	Prouds Road	OK
16A	87.528	R	Property		OK
16A	89.186	R	Paddock		OK
16A	92.52	L	Road	Biloela – Duaringa Road	OK
16A	92.52	R	Road	Biloela-Duaringa Road	OK
16A	94.051	R	Road	Cabarita Road	OK
16A	94.01	L	Property		OK
16A	94.281	L	Track	Gravel Track	OK
16A	94.468	R	Stopping Bay	Truck pull-off area	OK
16A	96.339	L	Road	Road to Baralaba	OK
16A	97.349	R	Property		OK
16A	98.029	R	Paddock		OK
16A	98.147	L	Track	Dirt Track	OK
16A	99.098	L	Track	Dirt/Gravel Track	OK
16A	100.601	L	Road	Laguna Road "Laguna"	OK
16A	103.855	R	Paddock		OK
16A	104.035	R	Paddock	Rural Address 10404	OK
16A	104.521	R	Paddock		OK
16A	105.201	R	Property	Monomeath Road. "Duaringa Selling Complex"	OK
16A	105.235	L	Other	Parthenium Weed Washdown Facility	OK

Appendix H.B – Attributes Before and After Segmentation

Appendix I – Linear Referencing

I.A – Attributes Before and After Linear Referencing

I.B – Linear Referenced Point Objects

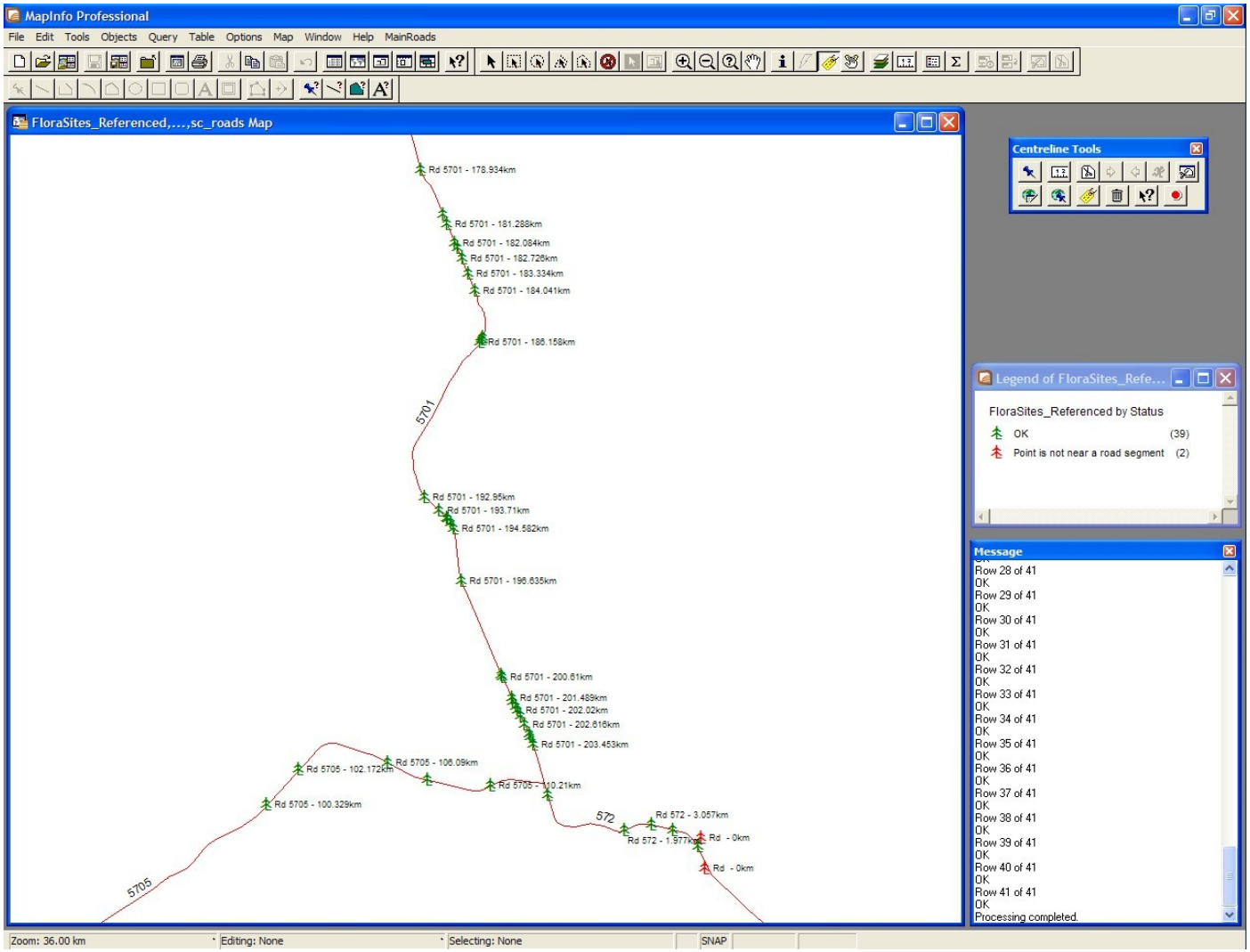
FloraSites Browser

CAPTURE_DATE	CAPTURE_DEVICE_ID	SIDE_OF_CARRIAGEWAY	DENSITY
26/05/2008	D07_E10_DCQ5	left	moderate
26/05/2008	D07_E10_DCQ5	left	moderate
26/05/2008	D07_E10_DCQ5	right	scattered
26/05/2008	D07_E10_DCQ5	left	scattered
26/05/2008	D07_E10_DCQ5	both	high
26/05/2008	D07_E10_DCQ5	left	scattered
26/05/2008	D07_E10_DCQ5	both	low
26/05/2008	D07_E10_DCQ5	left	scattered
26/05/2008	D07_E10_DCQ5	left	scattered
26/05/2008	D07_E10_DCQ5	left	high
26/05/2008	D07_E10_DCQ5	right	moderate
06/06/2008	D07_E12_DCQ2	right	scattered
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	left	low
06/06/2008	D07_E12_DCQ2	left	low
06/06/2008	D07_E12_DCQ2	both	scattered
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	left	low
06/06/2008	D07_E12_DCQ2	right	low
06/06/2008	D07_E12_DCQ2	right	low
06/06/2008	D07_E12_DCQ2	both	scattered
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	right	low
06/06/2008	D07_E12_DCQ2	right	low
06/06/2008	D07_E12_DCQ2	right	low
06/06/2008	D07_E12_DCQ2	left	low
06/06/2008	D07_E12_DCQ2	left	low
06/06/2008	D07_E12_DCQ2	both	scattered
06/06/2008	D07_E12_DCQ2	right	scattered
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	right	scattered
06/06/2008	D07_E12_DCQ2	both	scattered
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	left	low
06/06/2008	D07_E12_DCQ2	right	scattered
06/06/2008	D07_E12_DCQ2	left	low
06/06/2008	D07_E12_DCQ2	left	scattered
06/06/2008	D07_E12_DCQ2	left	low
26/05/2008	D07_E10_DCQ5	both	low

FloraSites_Referenced Browser

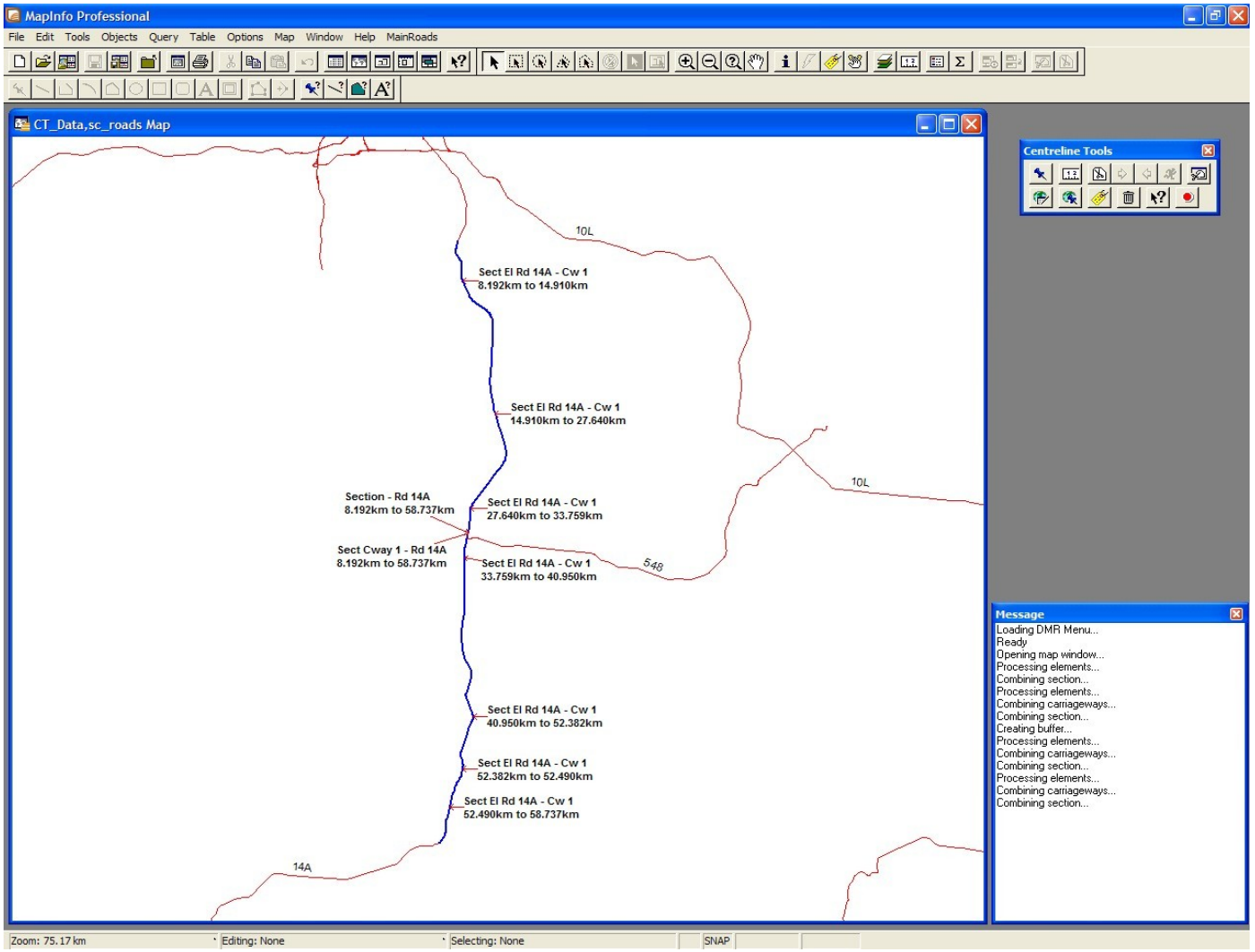
CAPTURE_DATE	CAPTURE_DEVICE_ID	SIDE_OF_CARRIAGEWAY	DENSITY	RoadID	Cway	Chainage	Status
26/05/2008	D07_E10_DCQ5	left	moderate	5705	1	110.21	OK
26/05/2008	D07_E10_DCQ5	left	moderate			0	Point is not near a road segment
26/05/2008	D07_E10_DCQ5	right	scattered	5705	1	100.329	OK
26/05/2008	D07_E10_DCQ5	left	scattered	5705	1	102.172	OK
26/05/2008	D07_E10_DCQ5	both	high	5705	1	106.09	OK
26/05/2008	D07_E10_DCQ5	left	scattered	5705	1	112.821	OK
26/05/2008	D07_E10_DCQ5	both	low	572	1	1.977	OK
26/05/2008	D07_E10_DCQ5	left	scattered	572	1	3.057	OK
26/05/2008	D07_E10_DCQ5	left	scattered	572	1	3.876	OK
26/05/2008	D07_E10_DCQ5	left	high	572	1	5.088	OK
26/05/2008	D07_E10_DCQ5	right	moderate			0	Point is not near a road segment
06/06/2008	D07_E12_DCQ2	right	scattered	5701	1	203.453	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	203.165	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	203.033	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	202.616	OK
06/06/2008	D07_E12_DCQ2	both	scattered	5701	1	202.257	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	202.02	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	201.863	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	201.638	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	201.489	OK
06/06/2008	D07_E12_DCQ2	right	low	5701	1	200.61	OK
06/06/2008	D07_E12_DCQ2	right	low	5701	1	200.521	OK
06/06/2008	D07_E12_DCQ2	both	scattered	5701	1	196.635	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	194.582	OK
06/06/2008	D07_E12_DCQ2	right	low	5701	1	194.337	OK
06/06/2008	D07_E12_DCQ2	right	low	5701	1	194.18	OK
06/06/2008	D07_E12_DCQ2	right	low	5701	1	194.102	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	193.71	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	192.95	OK
06/06/2008	D07_E12_DCQ2	both	scattered	5701	1	186.158	OK
06/06/2008	D07_E12_DCQ2	right	scattered	5701	1	186.028	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	185.922	OK
06/06/2008	D07_E12_DCQ2	right	scattered	5701	1	184.041	OK
06/06/2008	D07_E12_DCQ2	both	scattered	5701	1	183.334	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	182.726	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	182.267	OK
06/06/2008	D07_E12_DCQ2	right	scattered	5701	1	182.084	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	181.288	OK
06/06/2008	D07_E12_DCQ2	left	scattered	5701	1	180.895	OK
06/06/2008	D07_E12_DCQ2	left	low	5701	1	178.934	OK
26/05/2008	D07_E10_DCQ5	both	low	5705	1	107.745	OK

Appendix I.A – Attributes Before and After Linear Referencing



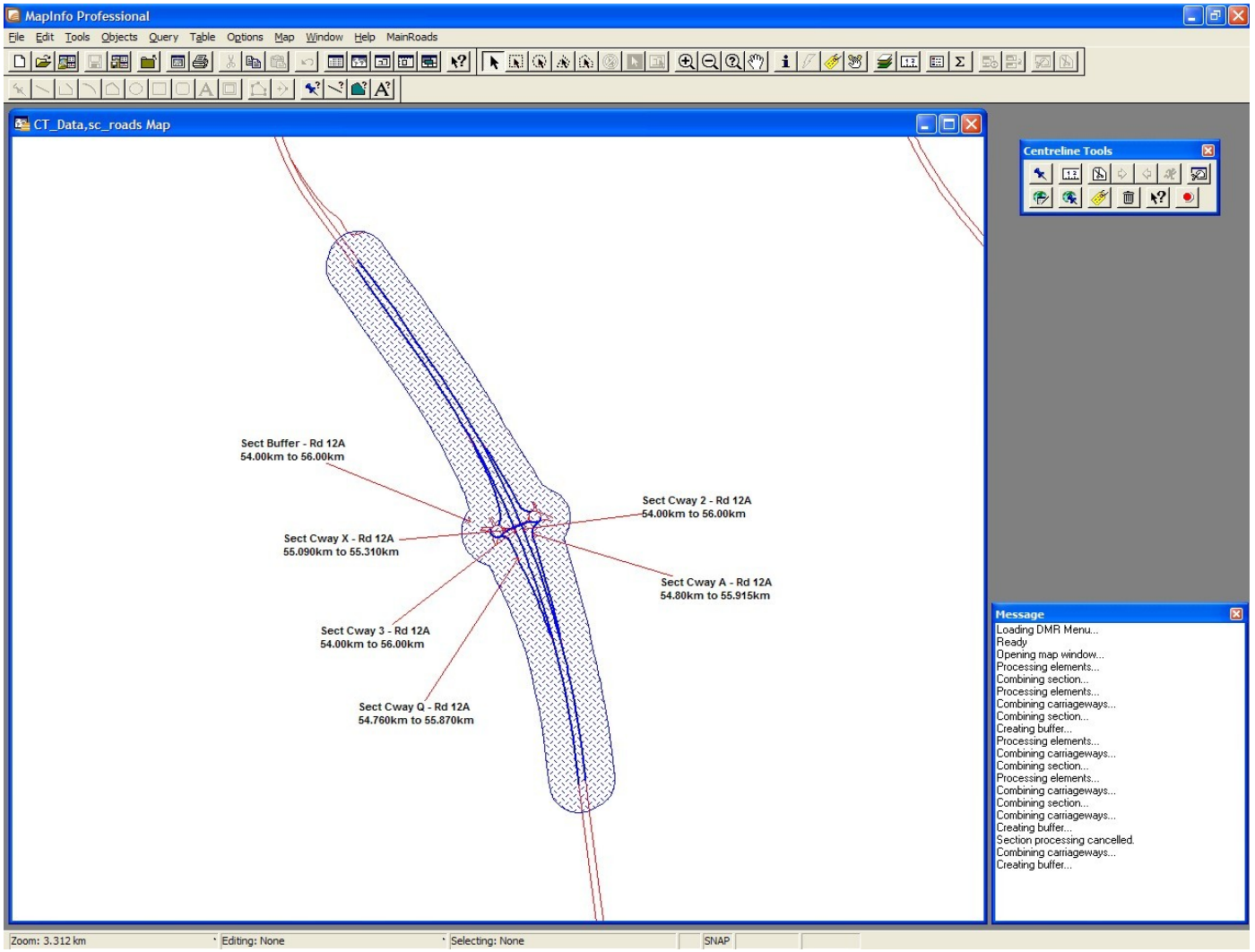
Appendix I.B – Linear Referenced Point Objects

Appendix J – Section Defined by Spatial Coordinates



Appendix J – Section Defined by Spatial Coordinates

Appendix K – Section Defined by Linear References



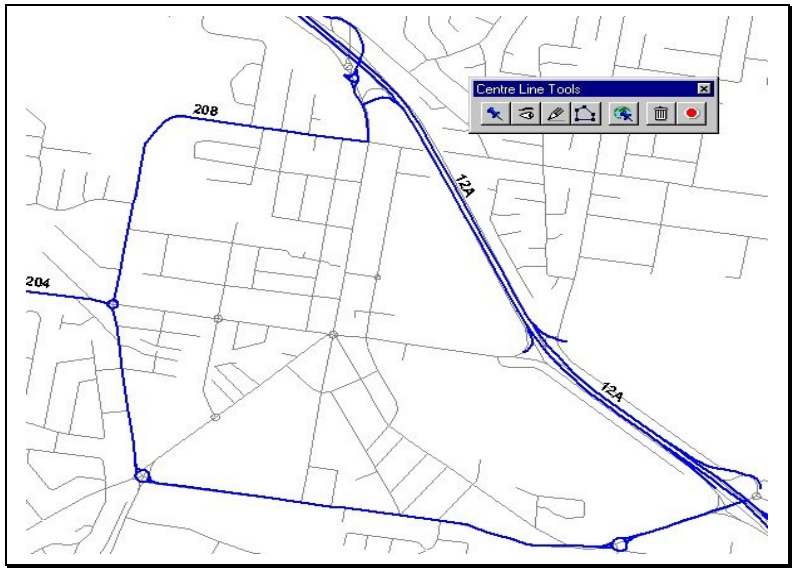
Appendix K – Section Defined by Linear References

Appendix L – Original Centre Line Tools User Guide

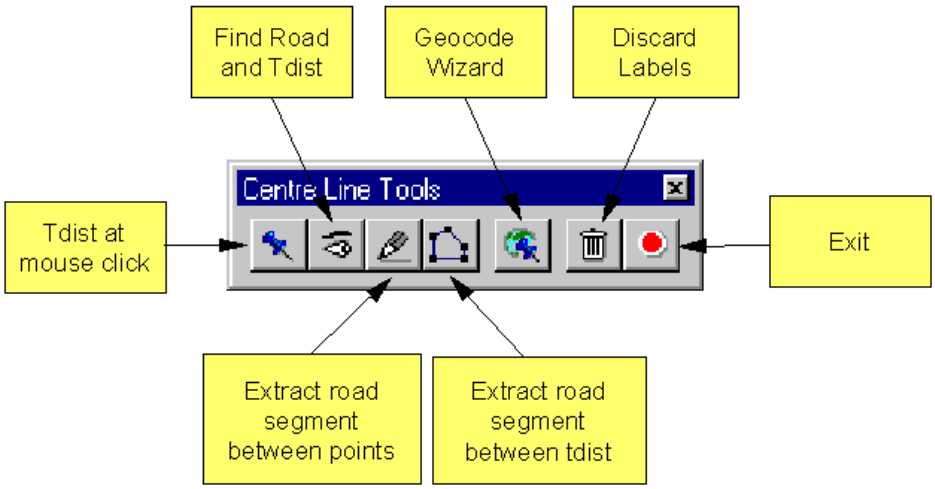
5 CENTRE LINE TOOLS

The Road Centre Line Tools is a suite of functions that allow users to perform several GIS operations on the state controlled network road centre line data. For example, it is possible for a user to view the map at a desired location described by the road number and the through distance. Also, by using the centre line tools it is easy to create new map objects of features using description information such as the road number, carriageway id, starting through distance, length of feature and width.

Note: Centre Line Tools is available to districts with a specially prepared digital road table. This table format is specific and further processing is required to produce a suitable input table for Centre Line Tools to function.

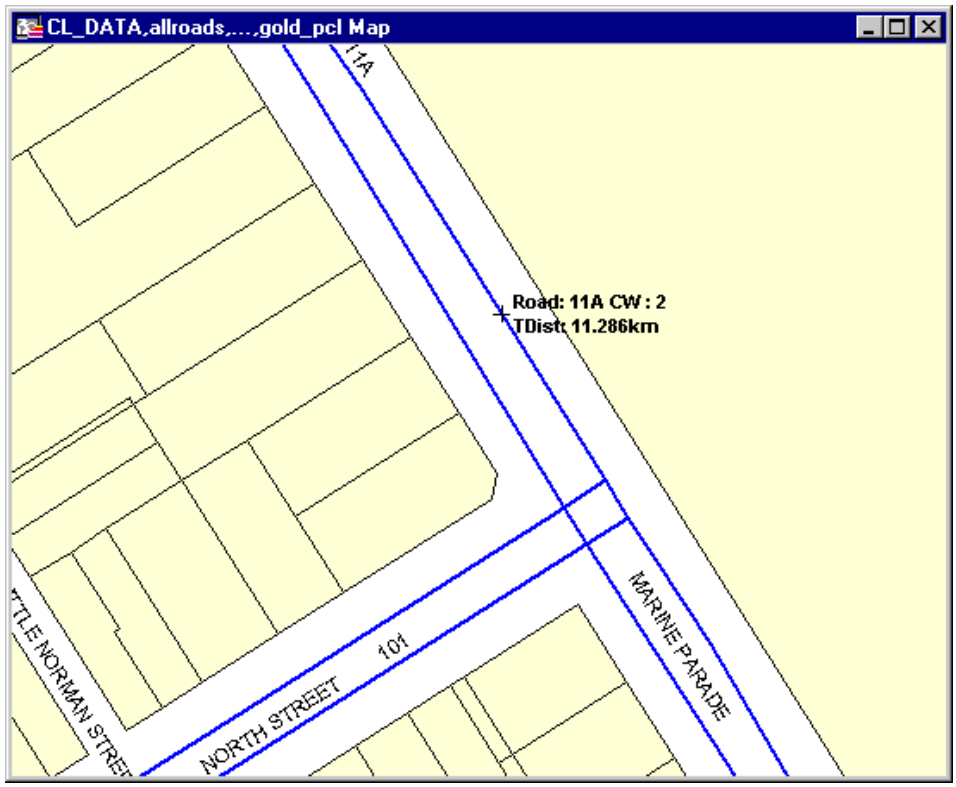


5.1 The Centre Line Tools Button Pad



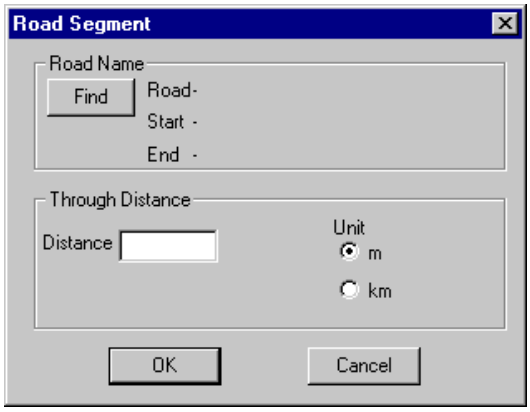
5.1.1 Tdist at Mouse Click 

This feature will return the road number, carriageway and through distance at the location where you click the mouse on a state controlled centre line, as in the example below.



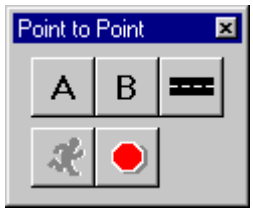
5.1.2 Find Road and Tdist 

This function enables you to find a location described by the road number and through distance. On selecting this feature the dialog below appears, requiring you to select the road from a list that is presented. After selecting the road enter the through distance. The map view will change to show the location you described and a label will appear at that point along the road, similar to the example above.



5.1.3 Extract Road Segment Between Points 

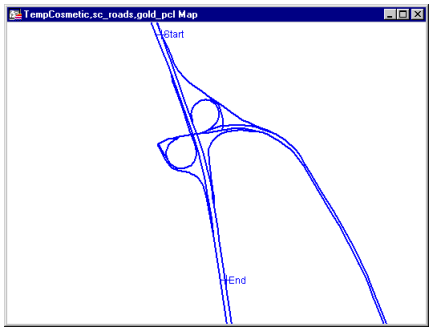
This feature allows the user to interactively extract a road segment identified by the start and end points along a road segment. Upon selecting this tool the "Point to Point" tool pad appears, as displayed below.



The "Point to Point" button pad allows you to select the start and end points, choose the road and process the query.

Select the start of the segment of interest, and

Select the end.



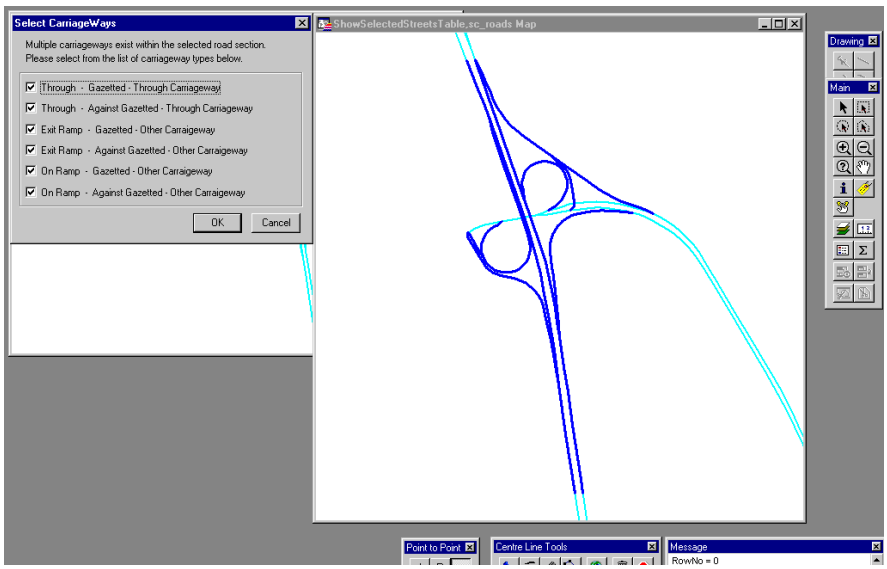
Example of start and end points selected.


Select the road to process.


Process the road segment.

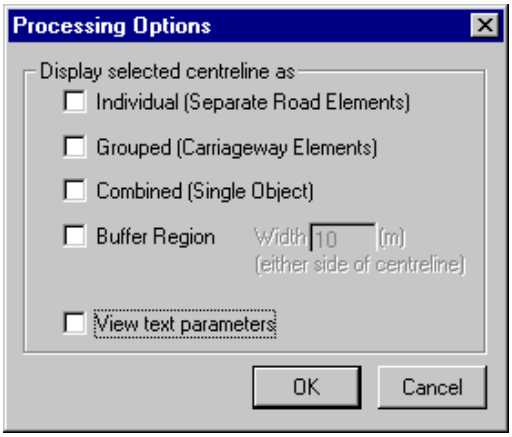

Cancel the point to point selection process.

When the selected road segment is processed you will be presented with a preview of the new road segment and a pick list of the carriageways within the selected road segment.

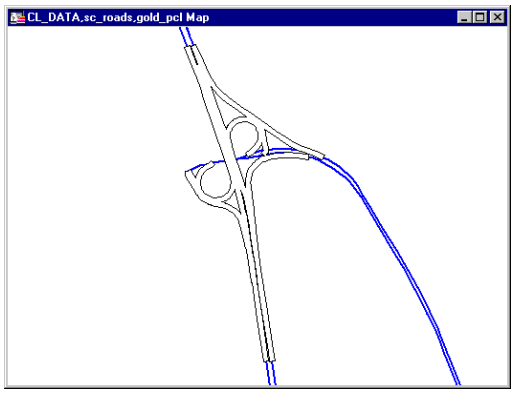


You can select all or any of the carriageways from the pick list to determine the desired output. As you alter the selection of carriageways the view of the road segment is updated to display the selected carriageway types.

After selecting the carriageway type you can select from any of the processing options. These enable you to represent the selected carriageway elements as individual polyline segments, as polylines grouped by carriageway, as a single combined object, or as a buffered region object.

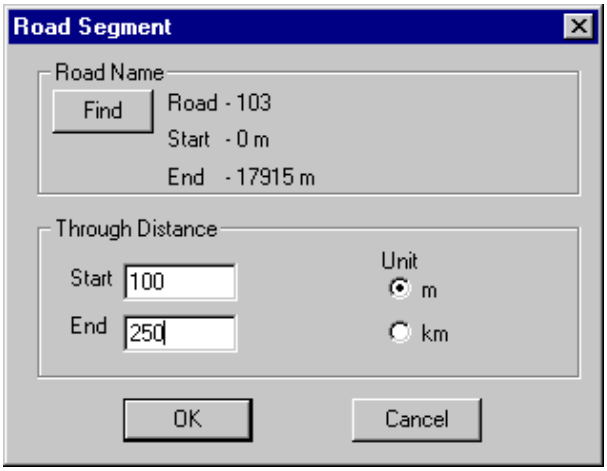


Below is an example of a 10 metre buffered object for the road carriageways selected on the previous page.

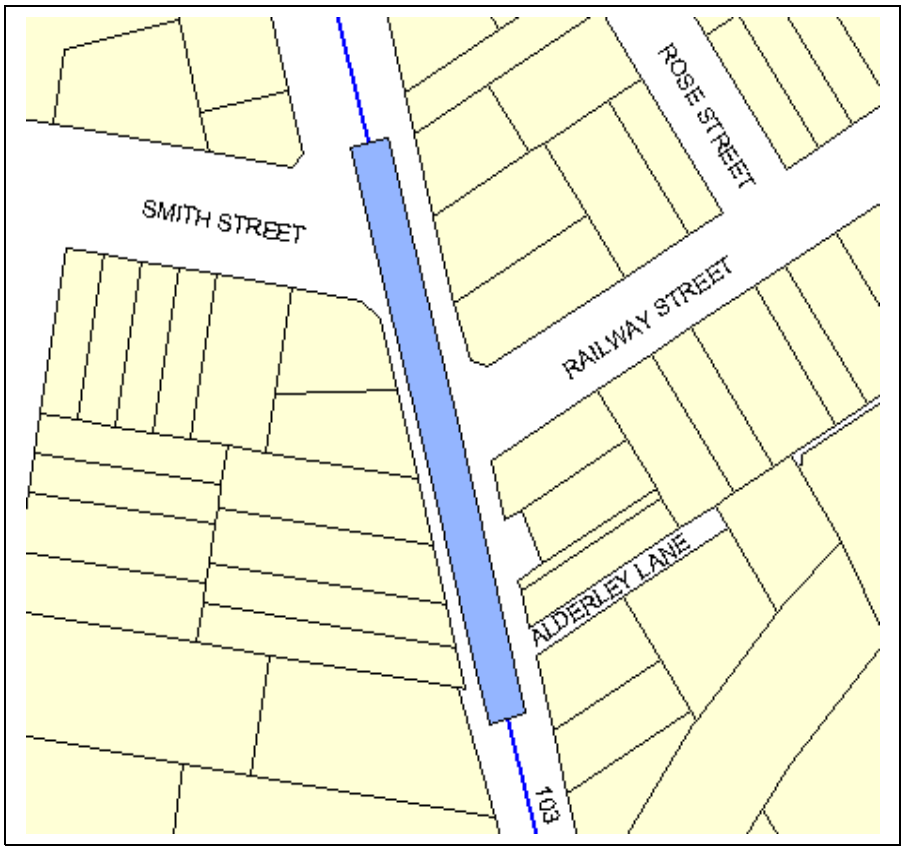


5.1.4 Extract Road Segment Between Tdist 

This tool is similar to the previous feature, except in this case you identify the required segment by entering the road number, carriageway and start and end through distances.



The extraction is processed exactly the same as the "Extract Road Segment Between Points" described beforehand. A selection list of carriageways will be presented followed by a list of processing options. Below is an example of a typical segment extraction.



5.1.5 Geocode Wizard 

The Geocode Wizard is a batch processing geocoder specifically for the State controlled road centre line. The system can geocode data (i.e. create map objects) using the ARMIS road reference description. The system can also determine the ARMIS road reference attributes for a geographic object along the state controlled road centre line.

5.1.5.1 Geocoding using road and tdist

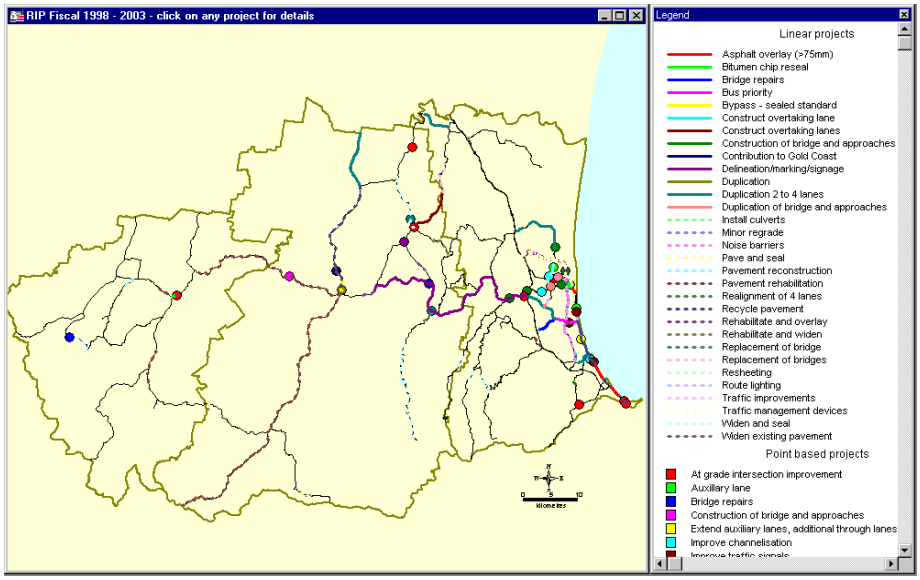
The application can process any MapInfo table containing road number and tdist descriptions. For example, a file containing the Road Implementation Program (RIP) data can have map objects created for it by using the textual location attributes such as the Road Number, Carriageway Identity and Through Distance. The resulting map and data can be viewed thematically and interrogated geographically.

ArmisData Browser

Road_Nc	CarrWay	Tdist	length	FeatType	Height	Width	datestr
102	2	0	0	ITS	0	0	27-JUN-97
102	3	10	0	PDC	0	0	27-JUN-97
102	3	1,530	0	ITS	0	0	27-JUN-97
102	2	1,530	0	ITS	0	0	27-JUN-97
102	2	3,280	10	CVT	0	0	27-JUN-97
102	2	3,290	0	ITS	0	0	27-JUN-97
102	3	5,130	10	CVT	0	0	27-JUN-97
102	2	5,130	10	CVT	0	0	27-JUN-97
102	3	5,220	10	CVT	0	0	27-JUN-97
102	2	5,220	10	CVT	0	0	27-JUN-97
102	2	5,390	120	BRG	99	8.5	27-JUN-97
102	3	5,620	10	CVT	0	0	27-JUN-97

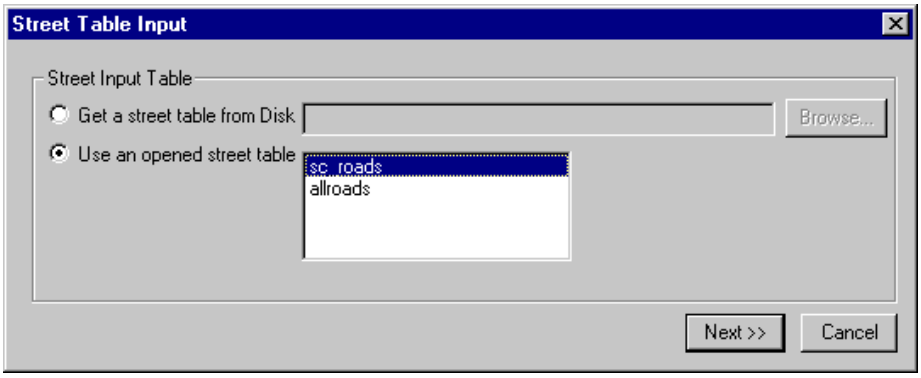
Example of some data suitable for geocoding using road reference description.

Below is a map and legend of the Road Implementation Program (RIP) for the South-Coast Hinterland District created using the Geocode Wizard.

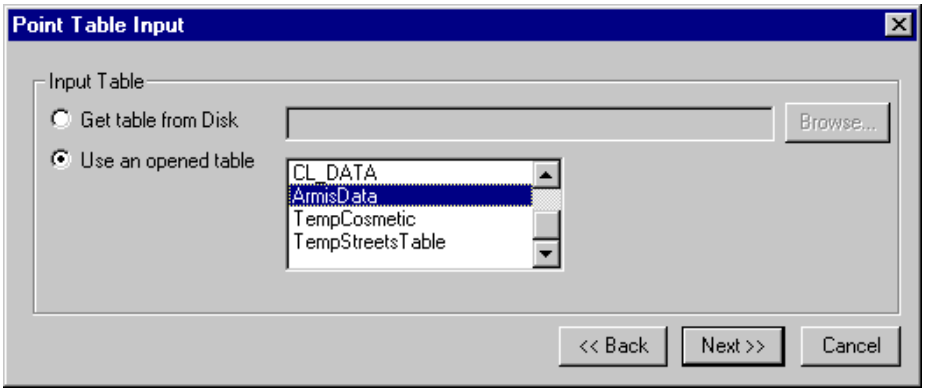


5.1.6 Procedure For Batch Geocode using Road ID and Tdist

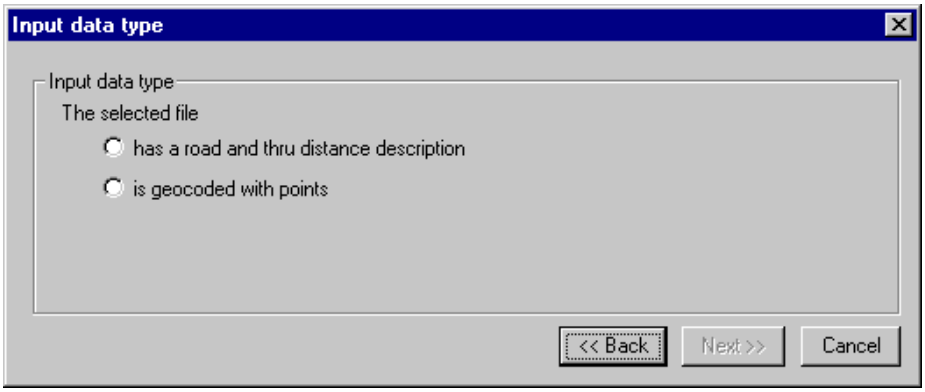
Step 1. From the Centre Line tools button pad, select the Geocode Wizard.



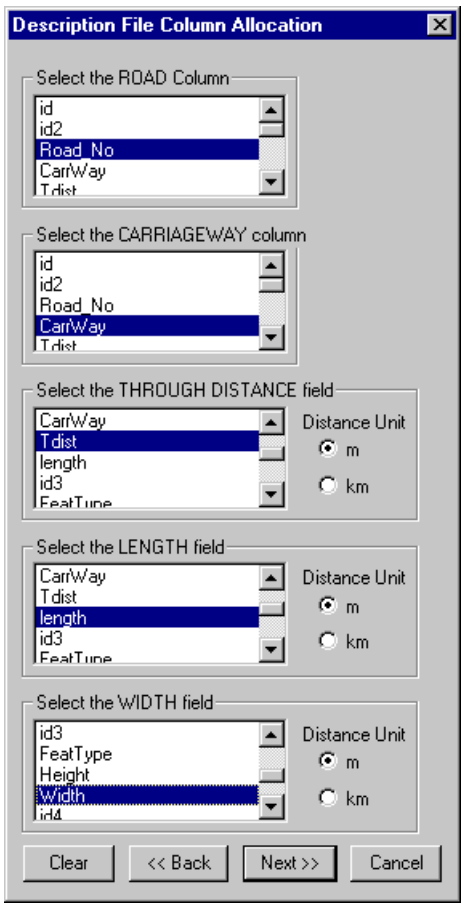
Step 2. Select the state controlled roads centre line street table, e.g. "sc_roads".
Step 3. Click on the "Next" button.



Step 4. Select the Input data table, (the table with tdist descriptions) either from the list, or choose browse to open the desired file.
Step 5. Click on the "Next" button.



Step 6. Select the radio button option "has a road and thru distance description"
Step 7. Click on the "Next" button.



Step 8. From the "Select the ROAD field" group click on the field name that contains the RoadID data.

Step 9. From the "Select the CARRIAGEWAY field" group click on the field name that contains the CARRIAGEWAY code data.

Step 10. From the "Select the THROUGH DISTANCE field" group click on the field name that contains the THROUGH DISTANCE code data, and select the distance units.

Step 11. From the "Select the LENGTH field" group click on the field name that contains the LENGTH code data, and select the distance units.

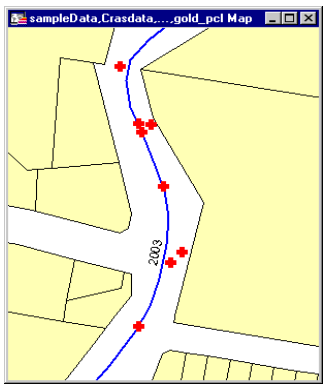
Step 12. From the "Select the WIDTH field" group click on the field name that contains the WIDTH data, and select the distance units.

Step 13. Click on the "Next" button.

The system will now process your file and create new map objects for each data record in your table. To view the results open the output file into a map window.

5.1.6.1 Determining road reference data for map objects

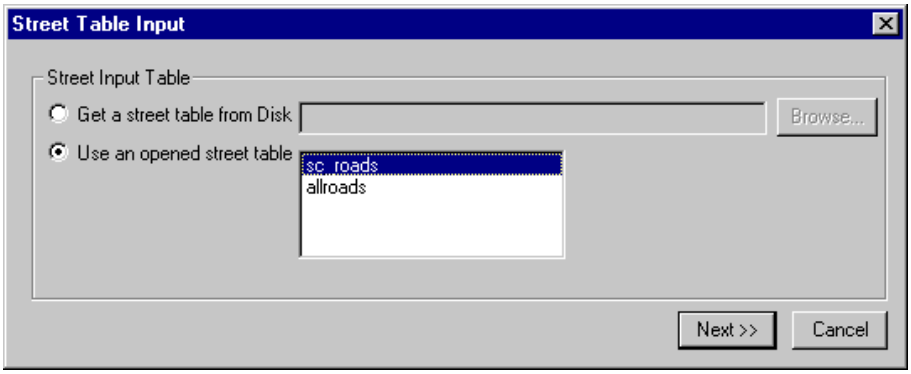
A table of map objects, such as points along a road, can have their road reference description determined using the Geocode Wizard. For example, a global positioning system could be used to create points locating road side assets in the field, the Geocode Wizard can be use to determine the road referencing information that Main Roads uses to describe those locations (Road number, Carriageway, and through distance).



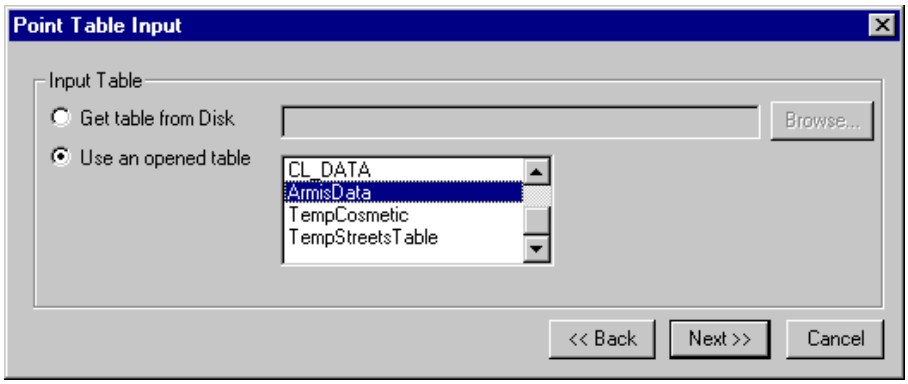
Example of a mapped table of point objects suitable for determining road reference data.

5.1.7 Procedure for determining road reference data for objects that are mapped

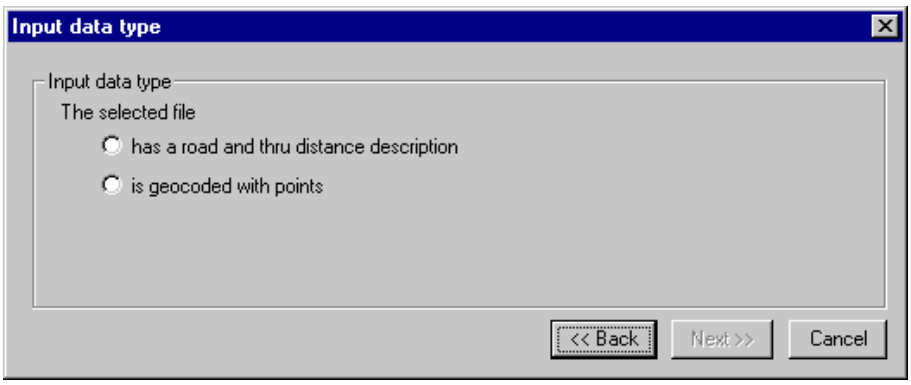
Step 1. From Centre Line tools button pad, select the Geocode Wizard.



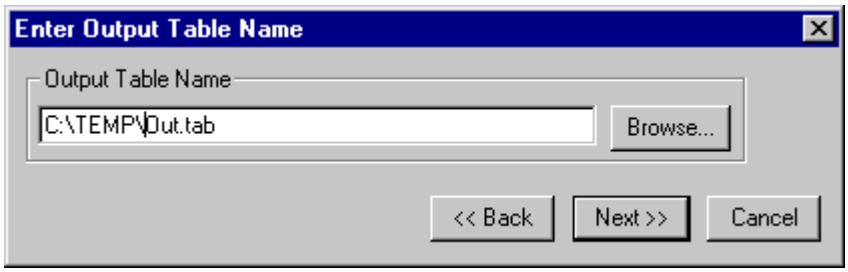
Step 2. Select the centre line street table, e.g. "sc_roads".
Step 3. Click on the "Next" button.



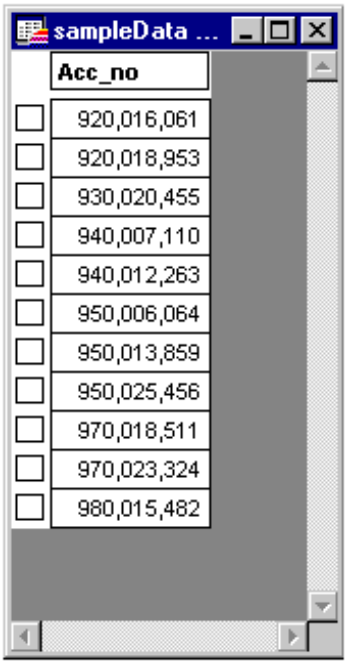
Step 4. Select the Input data table, either from the list or browse to open the file.
Step 5. Click on the "Next" button.

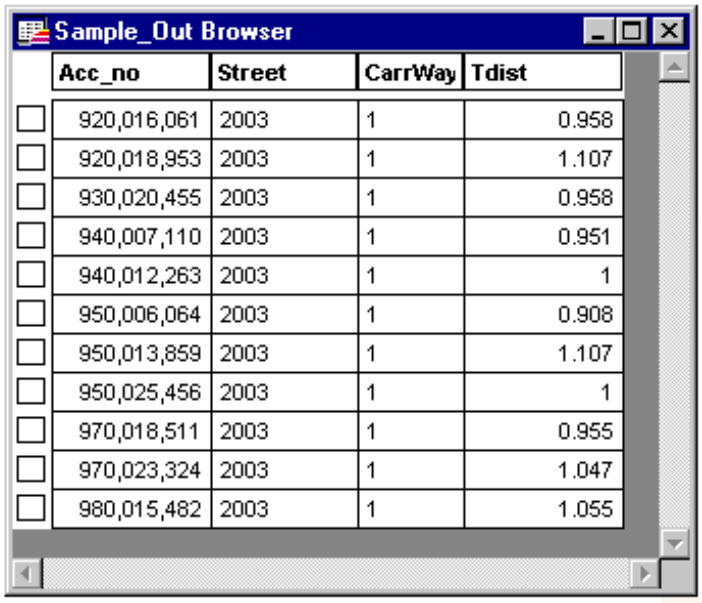


Step 6. Select the radio button option "is geocoded with points".
Step 7. Click on the "Next" button.
Step 8. Type, or browse for, an output file name.



The system will now process your table and produce an output table. Overleaf is an example of a typical input table and the resulting output table.





Acc_no	Street	CarrWay	Tdist
920,016,061	2003	1	0.958
920,018,953	2003	1	1.107
930,020,455	2003	1	0.958
940,007,110	2003	1	0.951
940,012,263	2003	1	1
950,006,064	2003	1	0.908
950,013,859	2003	1	1.107
950,025,456	2003	1	1
970,018,511	2003	1	0.955
970,023,324	2003	1	1.047
980,015,482	2003	1	1.055

5.1.8 Discard Labels 

Removes all map labels generated by the centre line tools system and clears the CL_DATA table.

5.1.9 Exit Centre Line Tools 

Closes Centre Line Tools and all files.