HEAVY LOAD VEHICLE ROUTING USING HIGHWAY NETWORK MODELS AND BRIDGE LOAD FORMULA

An interim report on

Study No. 2/10/5-91-1266

OVERWEIGHT PERMIT RULES BASED ON BRIDGE STRESSES

by

Roberto A. Osegueda
Civil Engineering Department
The University of Texas at El Paso

and

James S. Noel
Texas Transportation Institute
The Texas A&M University System

July, 1993

NOT INTENDED FOR CONSTRUCTION, BIDDING, OR PERMIT PURPOSES

TECHNICAL REPORT STANDARD TITLE PAGE

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
FHWA/TX-92 1266-3		

4. Title and Subtitle	5. Report Date
Heavy Load Vehicle Routing Using Highway Network Models and Bridge Load Formula	December, 1992 Revised: June, 1993
	6. Performing Organization Code
7. Author(s) Osegueda, Roberto A. and Noel, James S.	8. Performing Organization Report No. Research Report CE-92-2
9. Performing Organization Name and Address The University of Texas at El Paso Civil Engineering Department El Paso, Texas 79968-0516	10. Work Unit No. 11. Contract or Grant No. Study No. 2/10-5-91-1266
12. Sponsoring Agency Name and Address Texas Department of Transportation Transportation Planning Division P.O. Box 5051 Austin, Texas 78763	13. Type of Report and Period Covered Interim Report Sept. 1, 1990-Aug 31, 1992 14. Sponsoring Agency Code

15. Supplementary Notes

Research Performed in Cooperation with DOT, FHWA
Research Study Title: Overweight Permit Rules Based on Bridge Stresses

16. Abstract

A demonstration software for the routing of vehicles and for the retrieval of BRINSAP information of bridges along a route is presented. The software is demonstrated for its potential application in evaluating a proposed Bridge Load Formula. The software is implemented using a network model of the On-system roads of Tx Dot's District 12. The model was created from digitized maps and by defining node and link attributes. The definition of the bridges on the links and the nodes was accomplished by a mapping procedure and was verified using Tx Dot's Road Inventory sheets. Due to insufficient span length information in BRINSAP, implementation of the Bridge Load Formula requires determination of individual span lengths of all bridges.

17. Key Words	18. Distribution Statement
Bridges, Network Models, Digitized Maps, Coordinate Transformation	No restrictions. This document is available to the public through the National Technical Information Service, 5285 Port Royal Road, Springfield, Virginia 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	52	

Form DOT F 1700.7 (8-69)

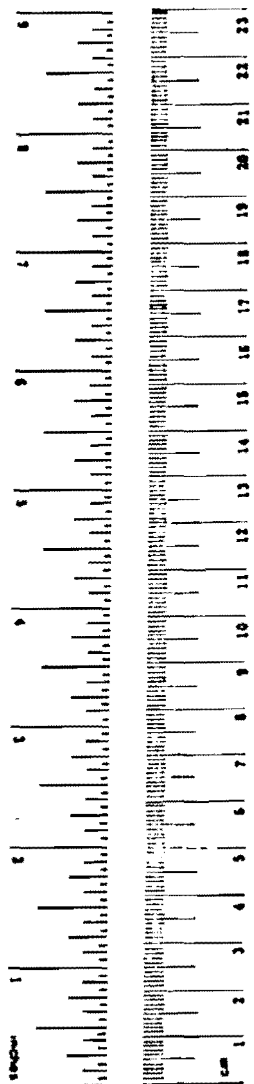
METRIC (SI*) CONVERSION FACTORS

APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
	LENGTH			
in	inches	2.54	centimetres	cm
ft	feet	0.3048	metres	m
yd	yards	0.914	metres	m
mi	miles	1.61	kilometres	km
	AREA			
in^2	square inches	645.2	centimetres squared	cm^2
ft^2	square feet	0.0929	metres squared	m^2
yd^2	square yards	0.836	metres squared	m^2
mi^2	square miles	2.59	kilometres squared	km^2
ac	acres	0.395	hectares	ha
	MASS (weight)			
oz	ounces	28.35	grams	g
lb	pounds	0.454	kilograms	kg
T	short tons (2000 lb)	0.907	megagrams	Mg
	VOLUME			
fl oz	fluid ounces	29.57	millilitres	mL
gal	gallons	3.785	litres	L
ft^3	cubic feet	0.0328	metres cubed	m^3
yd^3	cubic yards	0.0765	metres cubed	m^3

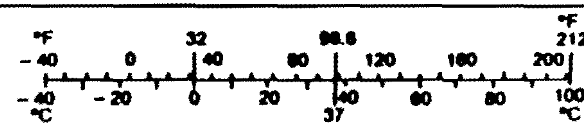
NOTE: Volumes greater than 1000 L shall be shown in m^3.

Symbol	When You Know	Multiply By	To Find	Symbol
	TEMPERATURE (exact)			
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C



APPROXIMATE CONVERSIONS TO SI UNITS

Symbol	When You Know	Multiply By	To Find	Symbol
	LENGTH			
mm	millimetres	0.039	inches	in
m	metres	3.28	feet	ft
m	metres	1.09	yards	yd
km	kilometres	0.621	miles	mi
	AREA			
mm^2	millimetres squared	0.0016	square inches	in^2
m^2	metres squared	10.764	square feet	ft^2
km^2	kilometres squared	0.39	square miles	mi^2
ha	hectores (10 000 m^2)	2.53	acres	ac
	MASS (weight)			
g	grams	0.0353	ounces	oz
kg	kilograms	2.205	pounds	lb
Mg	megagrams (1 000 kg)	1.103	short tons	T
	VOLUME			
mL	millilitres	0.034	fluid ounces	fl oz
L	litres	0.264	gallons	gal
m^3	metres cubed	35.315	cubic feet	ft^3
m^3	metres cubed	1.308	cubic yards	yd^3
	TEMPERATURE (exact)			
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



These factors conform to the requirement of FHWA Order 5190.1A.

* SI is the symbol for the International System of Measurements

ABSTRACT

A demonstration software for the routing of vehicles and for the retrieval of BRINSAP information of bridges along a route is presented. The software is demonstrated for its potential application in evaluating a proposed Bridge Load Formula. The software is implemented using a network model of the On-system roads of Tx Dot's District 12. The model was created from digitized maps and by defining node and link attributes. The definition of the bridges on the links and the nodes was accomplished by a mapping procedure and was verified using Tx Dot's Road Inventory sheets. Due to insufficient span length information in BRINSAP, implementation of the Bridge Load Formula requires determination of individual span lengths of all bridges.

DISCLAIMER

The contents of this report reflect the views of the authors who are responsible for the opinions, findings, and conclusions presented herein. The contents do not necessarily reflect the official views or policies of the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

TABLE OF CONTENTS

Chapter | Page

Chapter | Page

CHAPTER 1

INTRODUCTION, BACKGROUND AND OBJECTIVES

1.0 Introduction

A problem faced in U.S. highways is the constraint imposed on the transportation industry due to limitations of pavements, bridges and obstacles on the roads. These limitations typically constrain vehicles to minimum heights and widths, to minimum and maximum lengths (depending on the number of axles) and to a maximum allowable weight. On Texas roads, the maximum allowable weight, without a permit, is 80,000 lb. However, given current demands of industry, there are times when a truck must carry a load that exceeds the allowable. In this case, the individual responsible for the transport requests an overload permit.

1.1 Background of Tx DOT Permitting Process

The Central Permit Office (CPO) of the Tx Dot handles overload permit requests. Usually, if the load is slightly over the allowable, then the CPO grants the permit relatively fast. However, the CPO may dictate the route to follow. For unusually heavy vehicles, however, the trucker, when requesting a permit, must provide details of the equipment to be transported, the trailer to be used (number of axles, axle spacing, wheelbase, wheel gage, etc.), the origin and destiny, and the tentative route. The CPO then determines if the numbers of axles and tires are sufficient for not damaging the pavement. The information is then passed to the Division of Bridges and Structures for engineering evaluations of the bridges along the route. The bridge engineers evaluate each bridge on the proposed route and determine if it has sufficient strength to sustain the vehicle and its load. During the process, they identify all bridges to be crossed, retrieve the structural plans pertaining to critical bridges, and analyze them. The CPO will then approve or deny the permit requests based upon the capacity analyses performed. Furthermore, the CPO may require an alternate route, a maximum speed limit and/or a police escort.

The problems associated with the process of issuing overload permits for unusually heavy vehicles are as follows. First, the time it takes to process the permit requests is usually long. Second, the process requires a tremendous amount of engineering efforts. And third, permit fees are low when compared to the cost of issuing the permits.

1.2 Background of Bridge Formula

In an effort to ease the process of issuing overload permits, this project, conducted by the Texas Transportation Institute (TTI) and the University of Texas at El Paso (UTEP), was conducted. The overall objective was the development of a Bridge Load Formula and to demonstrate its application in the routing process of overload vehicles.

The formula was developed by the Texas Transportation Institute and its development is reported in Reference [1]. The proposed Bridge Load Formula is given by:

$$GW = \frac{1}{A + B*WB}$$

where,

$$A = 0.01884\ e^{(-0.009715\ L)}$$

$$B = 4.663x10^{-5} - \frac{0.00832}{L} - \frac{0.004034}{L^2}$$

L is the bridge span length in feet, WB is the wheelbase in feet if (WB < L), or WB is the bridge span length in feet if (WB > L), and GW is the group weight in kips. The group weight GW is the maximum allowable load of a group of axles. Since the proposed formula is a function of the span length(s) of the bridge and of the wheelbase of the vehicle, a route evaluation process would consist of identifying all bridges along a route, retrieving all span lengths and evaluating the formula for each span of each bridge. This would determine the allowable group weight corresponding to each bridge span. If the allowable group weight of a bridge is less than that of the actual vehicle, then an alternate route shall be considered bypassing the critical bridge or bridges. The work at U.T. El Paso consisted of investigating the implementation of such a formula by automating the identification of bridges along a given route, and the retrieval of bridge information from the Bridge Inventory, Inspection and Appraisal Program (BRINSAP) [2].

1.3 Objectives

The objective of this document is to report on the work accomplished at U.T. EL Paso as part of this project. The work centered in the development of a demonstration software to evaluate the bridges along routes for the incoming overload vehicles. The software is based on highway network models to simulate paths of travel and on the BRINSAP data base. The network models were created using digitized maps obtained from the Division of Transportation Planning. These drawings do not have all the identifications and the locations of the bridges. These were made part of the network model by mapping the geographic bridge coordinates stored in BRINSAP onto the digitized map. Then, the bridge identifications were attributed to the links of the network model. Thus, once identifying all segments of roads along a route, the bridges are identified and the corresponding BRINSAP records accessed. However, there is no room for leaving bridges out of the network model. For this reasons, the bridge locations within the network model were verified manually using maps made available by District 12 and using Tx Dot Road Inventory sheets.

1.4 Scope of the Research

The network models developed in this project correspond to the On-system highways of District 12 of the Texas highway system. No street, city or county roads were incorporated in the network model.

CHAPTER 2

ROUTING SOLUTION USING NETWORK MODELS AND BRINSAP

2.0 Introduction

Paths of travel are always determined using highway maps. If these maps had the exact location and identification labels of all bridges, then by a simple inspection, an individual can collect the identifications of all bridges to be crossed over. Then, each bridge file can be pulled out and the bridge information retrieved for evaluating and analyzing the bridge. A network model of the highway system and a data base file of updated bridge information can be used. First, a network model of a highway map can be created. This model may consist of nodes (simulating road intersections and/or interchanges) and links (simulating road segments between two intersections). Each node can have attributes such as the coordinates, the system and number of the highways being intersected, and others. Each link or road segment can have several attributes. These may include: route system, route highway number, length, control section numbers, number of bridges within the link, bridge identifications, bridge location, etc. Using a network model, all road segments (links) on a path of travel can be identified. Thus, if the bridge identifications are attributes of the road segments, then these are identified and the bridge characteristics can be extracted from the bridge data base. The Bridge Load Formula can then be evaluated to determine the adequacy of the bridge with respect to the overload.

2.1 Summary of Network Models

Network models are widely used for studying flow problems. They are used in areas such as electrical circuits, transportation, manufacturing processes, construction management, etc.. In this project, we were not interested in minimizing the path length between two points but to use the models for bridge identification and information retrieval purposes.

Any highway map can be modeled as a network. This has at least a set of nodes, representing highway intersections or interchanges, and a set of links, representing road segments. Figure 1 depicts an schematic of a network model. We see that there are many paths of travel between any two points.

Attributes that can be assigned to nodes and links are many. The selection of the attributes is determined by the information available. Attributes such as planar or geographic coordinates, identifications of links connecting to nodes, bridges on the nodes and others can be assigned to the nodes. Likewise, attributes that can be assigned to the links, besides connection points, are system type (Interstate, US, State Highway, Farm Market Roads, etc.), highway number, length of road segment, identification numbers of bridges on the link, direction of travel, etc.

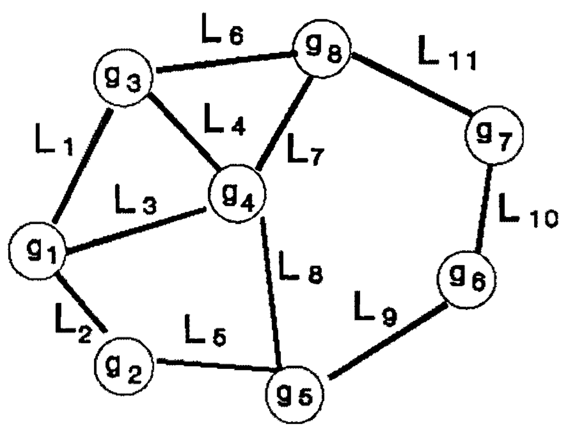


Figure 1 Schematic of a Network

The definition of the attributes also defines the scheme to follow for the route selection and for the bridge information retrieval. For route selection, intersection coordinates, information of the links connecting into the intersection, and the system type and highway number of the links, all can be used for selecting a route given a general path of travel. The general path of travel may consist of the principal routes and directions to be followed without specifying the individual road segments or links that are travelled. For example, a general path can be stipulated as: start at the intersection of IH-45 and FM 2553, go north on IH-45; take IH-10 west; take loop IH-610 north; and exit at the intersection of IH-610 and US-290. In this route, many intersections and road segments (links) are crossed. The identification of the segments can be accomplished by moving from node to node. At each node, the highways' numbers of the links intersecting it can be used to decide which link to continue on. Using this process, a complete list of links and nodes to be travelled can be compiled. Then, the bridge identification attributes of the links and nodes in this list, thus identify the bridges to be crossed. Then the process of retrieving the bridge information from BRINSAP is almost automatic. This approach requires that bridge attributes be assigned.

2.2 Information Available to Create Network Models

The information available to the authors to create a Network model for bridge information retrieval was the following: 1) BRINSAP files, 2) digitized maps of District 12, 3) Road Inventory Sheets of District 12, and 4) printed maps of District 12 with the bridge labels on.

2.2.1 BRINSAP

BRINSAP is the Tx Dot's program to implement the National Bridge Inspection Standards issued by the Federal Highway Administration [2]. BRINSAP is intended as a management tool rather than a data base of bridge details. It contains records of each bridge up to 179 fields. The following BRINSAP records were found useful in this research for locating bridges and to evaluate the bridge formula: control road ID and section number, structure number, route system, features crossed, facilities carried over, location, latitude and longitude coordinates, design load, safe load capacity, structure length, loading type, span type, number of spans and maximum span length. The fields such as control section number, location, features crossed, and latitude and longitude coordinates are helpful to locate a bridge and to assign the bridge identification attributes to the links of network models.

It should be noted, however, that BRINSAP fails to provide the lengths of the individual spans of a bridge. Since the Bridge Load Formula requires the span length, BRINSAP can only be used to retrieve the span length of simple-span and two-span bridges.

2.2.2 Digitized Maps

Tx DOT has implemented the use of Intergraph software for all drawings and maps they publish. Digitized geographic maps of the Texas highways currently exist and are available through the Division of Transportation Planning. The electronic maps are available as quad sheets; each quad sheet covers 7.5 degrees in latitude by 7.5 degrees by longitude. These maps use Lambert planar projection coordinates. To cover the entire District 12 more that 70 quad sheets were required. These maps have excellent accuracy but may fail to completely provide all details of road names, highways numbers, bridge locations, etc.. The maps include drawings elements indicating bridge locations, but we found that some bridges were not indicated.

2.2.3 Road Inventory Sheets

The Road Inventory Sheets (RI sheets) are Tx Dot's way to maintain an official inventory of the Texas highways. These consist of straight line diagrams of control sections showing bridge locations, roadway width, pavement type, etc. Initially, RI sheets were crated by draftpersons and were continuously updated as changes occurred on the roads. However, the process of updating information and maintaining an accurate inventory was cumbersome and time consuming. The Division of Transportation Planning is now adopting digitized road inventory sheets. In any event, the use of RI sheets is a way to ensure that all bridges be included in a network model.

2.2.4 Printed Maps of District 12

The District 12 offices has in its possession printed maps which include the bridge locations and labels. This information had been compiled as a means of managing their own bridges. In particular, these maps were very helpful to verify the locations of bridges in the network. Most importantly, however, was a sorting of identifications of the bridges on all complicated elevated interchanges. It should be remember that interchanges are considered as a single node. Thus, a node may have more than one bridge on it but not all the bridges will be crossed. The identifications of bridges on interchanges will be explained in a later section.

2.3 Tasks Faced

The challenges of this study primarily consisted in the manipulation of all the information required for the routing of vehicles and the retrieval of bridge identifications. Some of the tasks included: a) the creation of the network model by defining nodes and links and their corresponding attributes, b) the incorporation and accounting of all on-system bridges of District 12 into the network, c) modeling of interchanges and intersections, and d) the modeling of divided highways.

The creation of the network models was carried out semi-automatically using a CAD software and the geographic maps of the highways. First, all streets and city and county roads were deleted from the maps, leaving only the on-system roads. For each intersection or interchange (or node) of the highway map a unique identification number was assigned. The same was also done for the links or segments of the roads between two nodes. The geographical attributes that were considered critical to assign, were the node coordinates and the segment length. The node coordinates were determined using a macro program within the CAD software. This program asked the user to click on an intersection. Upon this action, the planar coordinates of the point were retrieved, and the node was assigned a unique number. This information was then permanently stored in a file. The length of the segments was similarly obtained using a macro program in the CAD software. The rest of the attributes were later assigned manually.

The incorporation of Bridges on Network Model was also a challenging task because it required that all bridges be accounted for. First, the BRINSAP coordinates were used to map the location of the bridges on the digitized maps. However, the BRINSAP coordinates are listed using latitudes and longitudes (spherical). The coordinate system of the digitized maps was a planar projection system (rectangular). Thus a transformation from latitude-longitude coordinates to rectangular coordinates was conducted. The procedure was reported in reference [3]. When the rectangular coordinates of the bridge location were available, a macro was program written within the CAD software to plot a drawing element at the bridge location and to label the corresponding bridge. About 95% of the bridges were mapped on the correct locations. For the other 5% of bridges, BRINSAP contains incorrect geographic coordinates or no coordinates at all. It was very clear however that most of the errors were made in keying the BRINSAP coordinates. The descriptions of the bridge locations were used to correctly position the bridges. All on-system bridges included in the BRINSAP file of District 12 that was available to U.T. El Paso were located on the digitized map.

The modeling of interchanges and intersections was done using a mini data base for each intersection. This data base included all possible paths of travel through an interchange or intersection. Each path contained a list the bridges to cross. Thus, if a node is found to be crossed, the path of travel is examined more carefully. For example, if the path of travel if from north bound lane of IH-45 to the west bound lane of IH-610, a different bridge will be crossed as if the path of travel were from the North bound of IH-45 to the east bound of IH-610.

Frontage roads were not considered in the model because it required the detailed modeling of all freeway exits. This activity was judged important but was not performed due to lack of time.

Divided Highways were modeled with parallel links. One link was specified to only carry traffic in one direction only. The parallel link was assigned to carry the flow of traffic in the opposite directions. The corresponding bridges were then assigned matching the direction of the traffic flow.

CHAPTER 3

DEVELOPMENT OF NETWORK MODEL

3.0 Introduction

This chapter provides details of the methods used to develop the network model of District 12. It includes the procedure implemented to attribute parameters to nodes (intersections) and links (road segments), and the procedure to attribute bridges to the corresponding links and nodes.

3.1 Summary of Procedure

It was explained in the previous chapter that a node is defined as the intersection of two or more roads, and a link is the road segment between two intersection points. The nodes and links were both given unique numbers to identify them. DBase III plus package and Intergraph's Microstation-PC software were used as tools for the generation of the data files of nodes and links.

3.2 Generation of Nodes and Node Attributes

The nodes of the network model have attributes as the planar coordinates, number of highways intersecting at the nodes, and the highway numbers which are intersecting etc. Microstation-PC software was used to generate the planar coordinates of the nodes. This software has a relational database interface. A macro program was written in Microstation-PC language to automatically generate the planar coordinates of nodes in a dBase III plus file by simply clicking at the node on the digitized map. This program first asked the user to input the node number for which he wants the planar coordinates. The program then asked the user to click at the node. When the user clicked, the microstation-PC retrieved the X and Y coordinates of that point. The user was then asked for the next node number and the procedure was repeated until all node coordinates were defined.

Another information attributed to the nodes was the intersecting highway details. This information was generated with a FORTRAN program which read the input from the link attributes, described in the next section, and outputs all the highways that are intersecting at each node.

3.3 Generation of Link Attibutes

The links of the network model have attributes as the two connecting nodes, highway type and highway numbers on which that particular links exist, the length of the link, the number of bridges on the link and the bridge identification numbers etc. Microstation-PC was used to determine the length along each link. First, the digitized maps were modified in the following manner. The digitized maps only consisted of drawing elements and did not necessarily started or ended at an intersection point. The drawing elements were broken up if they overlapped over nodes, and the elements were joined if there were two or more elements between two nodes. The modifications were needed to have only one drawing element per link or between two nodes. This way, a macro command of the Microstation PC language was used to retrieve the length of the links. A macro program was then written to retrieve the link length, to define the link number, the starting and ending nodes, the highway type (IH, US, etc.) and the highway number. All this information was stored in a Dbase III file.

Using this data file of link attributes, the highways that intersect at a node can were determined as mentioned in section 3.2. This was done with a separate FORTRAN program. For each node, the program reads the data file of link attributes and determines all links that connect into each node.

3.4 Bridges on Links and Nodes

As already explained, the bridges were mapped onto the digitized maps using coordinates transformed from longitude and latitude coordinates of the BRINSAP records. Each bridge included in BRINSAP was then attributed to the corresponding link(s) or node on which it exists. The bridges which lay on intersections or interchanges were attributed to the nodes. Separate files containing the bridge identification numbers of the links and the nodes were created. However, the fact that several bridges are identified to pertain to a particular node, the following logic was required to correctly identify the actual bridges to be crossed.

Consider the interchange of IH 610 WEST and US 59 SOUTH depicted in Figure 2. This figure includes the identification of all eleven bridges in the interchange. The following observations can be noted. If travelling North on IH 610 through the interchange, only bridge 0271-17-104 will be crossed. Likewise, if travelling South on IH 610 through the interchange, only bridge 0271-17-103 will be crossed. Travelling US 59 west onto IH 610 North, bridges 0027-13-427 and 0027-13-179 will be crossed. From IH 610 heading South onto US 59 East, bridge 0271-17-110 will be crossed, and so on. It can be seen that the particular bridges to be crossed depend on the path of travel within the interchange. For this reason, a mini data base was created for each interchange that contained the path of travel within the interchange and the corresponding bridges that will be crossed. This was done for all complicated interchanges. Thus, if a node is found to be crossed, the path is closely examined to determine the actual bridges.

3.5 Divided Highways

Divided Highways were considered by defining parallel links. In other words, a link modeling a divided highway has identical connecting nodes. One link is assign a direction of travel while the parallel link is assigned the opposite direction. Bridges on divided highways were also sorted and assigned to the link pertaining to the same sense of travel.

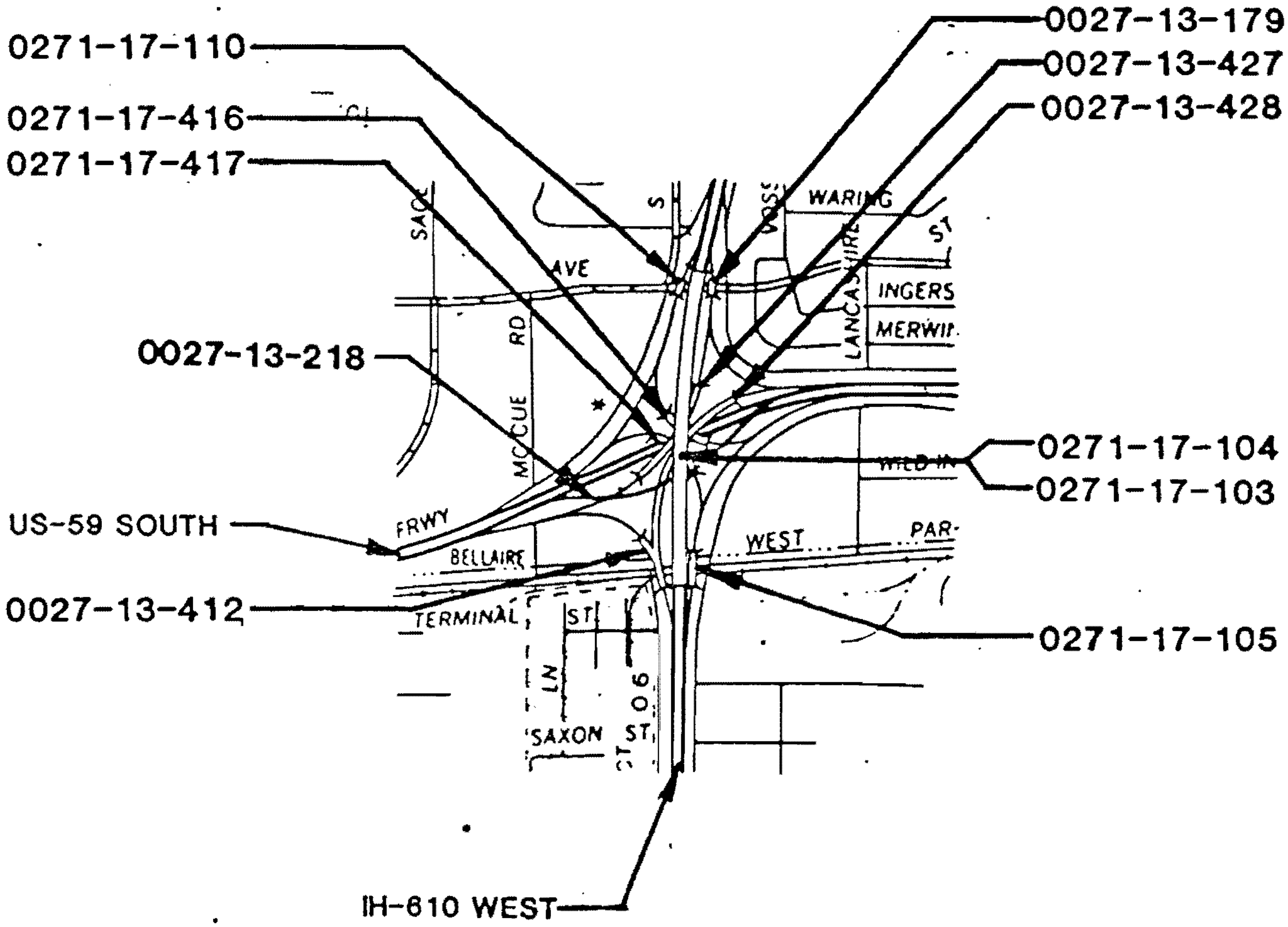


Figure 2. Details of IH 610 West and US 59 South Interchange.

CHAPTER 4

DESCRIPTION OF ROUTING SOFTWARE

4.0 Introduction

The material described in Chapters 2 and 3 summarized the activities, tasks and work accomplished for the implementation of a routing program that identifies bridges along a route. This chapter provides a brief description of the program. The program is intended to be used with a regular Texas Highway map without knowledge of the internal structure of the network model. The program was implemented using the network model of the On-System highways of Tx Dot District 12. Furthermore, due to the fact that the only span information BRINSAP includes is the major span length besides the total length of the structure, it is impossible to evaluate the proposed Bridge Load Formula for all bridges along a given route. For this reason, calculations of the allowable axle group weight for a given bridge using the formula were excluded from the program. It is the opinion of the authors that it should only be implemented when the span length information of all bridges be complete. This information however, may be available in the near future, since it may not be so difficult to obtain.

Appendix I shows a listing of the routing program BRG. This program was coded in Microsoft FORTRAN and executes in the PC environment. It requires a 80286 Personal Computer with math coprocessor and a minimum of 512 of free RAM memory.

4.1 Use of BRG program

The program is initiated by typing BRG and then the "Enter" key from the subdirectory that contains the executable files. First, the program will prompt you to enter a name for an output file. This file will be used to store the information on the route of travel and of the bridges on the route. The user needs to specify if he wants a short or a long output. It is recommended that the short output option be always selected. You will now see a message displayed in the computer screen informing the user about valid entries for highway types. These entries are IH for Interstate highway, US for US highway, SH for State highway, FM for Farmer Market road, SPUR, BW for Beltway, PR for Park Road and TOLL for Toll roads. A highway is identified by first entering the highway type, and then the highway number.

The next part corresponds to the input of the origin and destination of travel. It should be pointed out that travel is only modeled from a node to another node. Thus, the input and origin and destination is specified by providing two intersecting roads of the node. Proceeding is the input of the route selection. This is entered by specifying the highway identifications to be travelled plus the directions (N, E, S or W). To finish specifying the route simply enter END. It should be noted that if the incorrect information is entered, the program will alert you of that. Upon a successful input of the route selection, the computer will summarize your route request. Please verify if this information is correct. The program will then proceed to read the network information of District 12. You will see that the network model consists of 502 nodes, 738 links and 1,929 bridges. The program will now proceed to identify all links and nodes on the requested route and will conclude by displaying messages such as the number of nodes and links to be crossed, link and node numbers to be crossed, the length of travel, the number of bridges along the route, and the bridge labels of all bridges to be crossed. The program will now access the BRINSAP file and output file will be created providing with some of the characteristics of the bridges.

4.2 Sample Route

Consider the scenario that an overload permit is requested to move a piece of machinery from Ellignton Field to Cullen Blvd. near IH 610. By observing a normal Texas highway map, it can be seen that the closest on-system intersection to Ellignton Field is that of SH 3 and FM 1959. That to Cullen Blvd is IH 610 and FM 865. A tentative route is FM 1959 South, IH 45 North and IH 610 West. When this information is entered, the following is part of the screen display.

NUMBER OF POTENTIAL BRIDGES TO BE CROSSED 31

LENGTH OF TRAVEL PATH (IN MILES) 14.985250

BRIDGE IDENTIFICATION NOS THAT ARE CROSSED :

0271-16-361	0271-16-362	0271-16-364	0271-16-366
0271-16-482	0271-16-483	0271-16-486	0271-16-489
0271-16-493	0271-16-494	0271-16-495	0271-16-496
0500-03-014	0500-03-015	0500-03-021	0500-03-022
0500-03-023	0500-03-024	0500-03-025	0500-03-042
0500-03-054	0500-03-061	0500-03-073	0500-03-090
0500-03-091	0500-03-126	0500-03-208	0500-03-264

The information above includes the number of bridges to be crossed, the length of travel and the BRINSAP bridge labels. The length of the path is obtained by adding the individual length of all links.

This route and many other routes have been tested and have been verified by comparing the results to the bridges labels of the Printed Maps of District 12. So far all information seems to be working satisfactorily.

4.3 Software Disclaimer

Installation and official utilization of this software by personnel of Texas Department of Transportation may be prohibited by the Automated Division of Tx Dot. This software is only intended to demonstrate the route selection and bridge information retrieval process for a possible evaluation of the proposed Bridge Load Formula.

CHAPTER 5

SUMMARY CONCLUSIONS AND RECOMMENDATIONS

5.0 Summary

This report has summarized the activities accomplished by U.T. El paso under this joint project with the Texas Transportation Institute. The overall objective was the development of a Bridge Load Formula and to demonstrate its application in the routing process. The proposed formula was developed by the Texas Transportation and is reported in Reference [1]. Since the proposed formula is a function of the span length(s) of the bridge and of the wheelbase of the vehicle, a route evaluation process requires identifying bridges along a route and retrieving all span lengths. The work at U.T. El Paso consisted of investigating the implementation of the formula by automating the identification of bridges along a given route, and the retrieval of bridge BRINSAP.

This report also presented the development of a demonstration software to evaluate bridge along routes for incoming heavy load vehicles. The software is based on highway network models to simulate paths of travel and on the BRINSAP data base. A network model of Tx Dot District 12 was created using digitized maps. Bridge labels were made part of the network model by mapping the geographic bridge coordinates onto the maps. The software identifies all segments of travel along a route with their corresponding bridges. The calculations of the Bridge Load Formula were left out of the software because BRINSAP lacks sufficient information to determine the individual span lengths of multispan bridges. However, if this information were collected, then it would not be too much of a problem in implementing the formula.

5.1 Conclusions

This investigation has laid out a methodology for the routing of heavy load vehicles to minimize bridge stress based upon a Load Formula. It is concluded that an automated implementation of a bridge load formula to evaluate bridges along routes of heavy-load vehicles is practical. This implementation will require network models of our highways, an accurate accounting of all bridges and additional span information of all bridges. The routing software has demonstrated that savings could result if this or a similar software be implemented.

5.2 Recommendations

It is recommended that the Texas Department of Transportation consider upgrading the information stored in BRINSAP. It is suggested that the individual span lengths of all bridges be included. Another recommendation is the creation of a separate bridge details information system that would allow engineers the access to all bridge details electronically. It this were to exist, the evaluation of incoming vehicles could be done by an expert system performing automated analysis on the individual bridges. This way, uncertainties could be eliminated.

REFERENCES

1. James S. Noel, et. al. Research Report, 1266-1, Texas Transportation Institute, The Texas A&M University System, College Station, Texas. (not published yet).

2. Bridge Inventory, Inspection and Appraisal Program - Manual of Procedures, Texas Department of Transportation, Division of Safety and Maintenance Operation, September 1984, Austin, Texas.

3. R.A. Osegueda, et. al., Towards the Implementation of an Overload Permit Formula Using Network Models and BRINSAP

APPENDIX

LISTING OF BRG PROGRAM

This program was coded and compiled using Microsoft Fortran version 5.0. The program requires a math co-processor and a minimum of 512 KB of free RAM space. For an executable copy of this program, please contact the author at the address listed in the cover sheet. The authors are not responsible for any use of this software. It is only intended to be a demonstration software for possible implementation of the proposed Bridge Load Formula

LISTING

```
$debug
      include 'fgraph.fi'
      include 'fgraph.fd'
      record /rccoord/ curpos
      INTEGER I,J,K,kode,nseg,node(600),link(1200),nc(1200,2),nn,nl,
     & nb(1200),ibr(1100,50),idb(4000),nbatt,itemp,NLNOD(2000),
     & NOD,LNUM(1110,6),LTRAVEL(100),NTRAVEL(100),NODE1,NODE2,
     & ndef(1100)
      INTEGER ISTR(200,40),in(20)
      real x(600),y(600),xl(1200),XXL,CO,SI
      CHARACTER*1 DIR(100), ans,str1
      character*4 str4
      character*6 str6,CTSEC
      character*10 str10
                  CHARACTER*9 CIDB1,cidb2,cidb3,cidb4
                  CHARACTER*3 CTSECSTR
      CHARACTER A*514
      CHARACTER*4 HWTYP(100),HWNO(100),NODHWN(600,3),nodhwt(600,3),
     &  lhwt(800,3),lhwn(800,3)
      character*4 hwtp1,hwno1,hwtp2,hwno2
      CHARACTER COUNTY*3,CONT*4,SEC*2,MP*5,STNUM*3,RDES*1,RS*2,RN*4
      CHARACTER RDIR*1,FCROS*24,CRBR*1,FOVER*18,LC*25,VCLEAR*4
      CHARACTER FCLASS*2,YB*4
      CHARACTER LANON*2,LANUN*2,DLOAD*1,APW*3,OPST*1,LTYPE*1
      CHARACTER LOAD*3,TYSER*2
      CHARACTER MSTYP*4,MJTYP*4,MNTYP*4,NMSPAN*3,NMJSPAN*3,NMNSPAN*3
      CHARACTER TOTSPAN*4
      CHARACTER HCLEAR*4,MXSPAN*4,STRLEN*6,RWIDE*4,DCKWID*4
      CHARACTER DCKCOND*1,SPRCOND*1
      CHARACTER SBCOND*1,OPRAT*3,IRATING*3,STREVA*1,SLCAP*1,DEFHD*1
      integer STNODE,ENDNODE
      integer*2 dummy2, row
      integer*4 dummy4
      character*12 nout
      character*18 descnode
      CHARACTER*4 C1,C4,C7,C10
                  CHARACTER*2 C2,C5,C8,C11
                  CHARACTER*3 C3,C6,C9,C12

      dummy4 = setbkcolor(int4(0))
      call clearscreen($gclearscreen)
      dummy4 = setbkcolor(int4(4))
      dummy2 = settextcolor(int2(15))
      call settextposition(int2(23),int2(1),curpos)
```

```
      call outtext(' ENTER the name of the output file: ')
      READ(*,'(a12)') nout
      OPEN(1,file = nout, status='new')
      OPEN(2, file = 'd12lk.txt',STATUS='OLD')
      OPEN(3, FILE='d12hw.out',status='old')
      open(4,file='d12lkbrd.txt',status='old')
      open(6,file='lkmuldef.txt',status='old')
      OPEN(11,FILE='INTCHG.TXT',STATUS='OLD')
      ipcode = 0
      do while (ipcode.lt.1.or.ipcode.gt.2)
      dummy4 = setbkcolor(int4(0))
      call clearscreen ($gclearscreen)
      dummy4 = setbkcolor(int4(1))
      call settextposition(INT2(2),INT2(23),curpos)
      call outtext(' BRIDGE INFORMATION OUTPUT FORMAT ')
      call settextposition (int2(4),int2(30),curpos)
      call outtext(' (1) Short Output ')
      call settextposition (int2(5),int2(30),curpos)
      call outtext(' (2) Long Output ')
      dummy4 = setbkcolor(int4(4))
      call settextposition (int2(7),int2(23),curpos)
      call outtext(' Please enter your choice number: ')
      read(*,*) ipcode
      end do
C
C     This part takes the input related to the route to be travelled
C
      dummy4 = setbkcolor(int4(0))
      call clearscreen ($gclearscreen)
      dummy4 = setbkcolor(int4(1))
      call settextposition(INT2(2),INT2(15),curpos)
      call outtext(' BRIDGE AND ROUTE INFORMATION RETRIEVAL PROGRAM ')
      call settextposition(INT2(3),INT2(15),curpos)
      call outtext(' *******************************************   ')
      call settextposition(INT2(4),INT2(15),curpos)
      call outtext('                                  ')
      call settextposition(INT2(5),INT2(15),curpos)
      call outtext(' This program requires you to enter a starting ')
      call settextposition(INT2(6),INT2(15),curpos)
      call outtext(' node of origin, an ending node of destiny, and ')
      call settextposition(INT2(7),INT2(15),curpos)
      call outtext(' the highway identifications and directions to ')
      call settextposition(INT2(8),INT2(15),curpos)
      call outtext(' be followed.                            ')
      call settextposition(INT2(10),INT2(15),curpos)
      call outtext(' To define the highway identifications you need ')
      call settextposition(INT2(11),INT2(15),curpos)
      call outtext(' to specify highway type, and highway number. ')
      call settextposition(INT2(12),INT2(15),curpos)
      call outtext(' Valid entries for the highway type consist of: ')
      call settextposition(INT2(13),INT2(15),curpos)
      call outtext(' I    Interstate highway               ')
      call settextposition(INT2(14),INT2(15),curpos)
      call outtext(' US   US highway                       ')
      call settextposition(INT2(15),INT2(15),curpos)
      call outtext(' SH   State highway                    ')
      call settextposition(INT2(16),INT2(15),curpos)
      call outtext(' FM   Farm Market Road                 ')
      call settextposition(INT2(17),INT2(15),curpos)
      call outtext(' SPUR Spur highway                     ')
      call settextposition(INT2(18),INT2(15),curpos)
      call outtext(' BW   Beltway                          ')
      call settextposition(INT2(19),INT2(15),curpos)
      call outtext(' P    Park Road                        ')
      call settextposition(INT2(20),INT2(15),curpos)
      call outtext(' TOLL Toll Road                        ')
      call settextposition(INT2(22),INT2(15),curpos)
      call outtext(' USE CAPITAL LETTERS ONLY !            ')
      call settextposition(INT2(24),INT2(40),curpos)
```

```
      call outtext(' hit any key to continue ')
      pause ' '
      WRITE(1,*) 'BRIDGE AND ROUTE INFORMATION RETRIEVAL PROGRAM'
      WRITE(1,*) '********************************************'
      WRITE(1,*) ' '
      write(1,*) '  This program requires you to enter a starting node'
      write(1,*) '  of origin, an ending node of destiny, and the '
      write(1,*) '  highway identifications and directions to be '
      write(1,*) '  followed.'
      write(1,*)
      write(1,*) '   To define the highway identifications you need to '
      write(1,*) '   specify highway type, and highway number.'
      write(1,*) '   Valid entries for the highway type consist of:'
      write(1,*) '   I     Interstate highway'
      write(1,*) '   US    US highway'
      write(1,*) '   SH    State highway'
      write(1,*) '   FM    Farm Market Road'
      write(1,*) '   SPUR  Spur highway'
      write(1,*) '   BW    Beltway'
      write(1,*) '   P     Park Road'
      write(1,*) '   TOLL  Toll Road'
      write(1,*) ' '
 54   continue

c######################################################################
c
c      read node information
c
      IF (KREAD.EQ.0) THEN
      k = 1
 27   read(3,fmt=26,end=29) node(k),x(k),y(k),nodhwt(k,1),nodhwn(k,1),
     & nodhwt(k,2),nodhwn(k,2),nodhwt(k,3),nodhwn(k,3)
c     write(*,26) node(k),x(k),y(k), nodhwt(k,1),nodhwn(k,1),
c    &  nodhwt(k,2),nodhwn(k,2),nodhwt(k,3),nodhwn(k,3)
      k = k+1
      goto 27
 29   continue
      REWIND(3)
 26   format(i5,f16.5,f16.5,2x,a4,1x,A4,3x,a4,1x,A4,3x,a4,1x,A4)
      nn = k-1
      KREAD=1
      END IF


C     write(*,*) 'Enter the intersecting highwaytype1'
C     read(*,3) hwtp1
C     write(*,*) 'Enter the intersecting highwayno1'
C     read(*,3) hwno1
C     write(*,*) 'Enter the intersecting highwaytype2'
C     read(*,3) hwtp2
C     write(*,*) 'Enter the intersecting highwayno1'
C     read(*,3) hwno2

101   j=1
      dummy4=setbkcolor(int4(0))
      call clearscreen($gclearscreen)
      dummy4=setbkcolor(int4(1))
      call settextposition(3,int2(18),curpos)
      call outtext(' ENTER INTERSECTING HIGHWAYS FOR START POINT  ')
      call settextposition(6,int2(18),curpos)
      call outtext(' HW1_TYPE  HW1_NO          HW2_TYPE   HW2_NO')

7      dummy4=setbkcolor(int4(0))
      col=j+20
      dummy4 = setbkcolor(int4(4))
      call settextposition(8,int2(col),curpos)
      call outtext('    ')
      call settextposition(8,int2(col),curpos)
        kode = 0
```

```
      read(*,3) HWTP1
      if (hwtp1 .eq. 'IH') kode = 1
      if (hwtp1 .eq. 'US') kode = 1
      if (hwtp1 .eq. 'SH') kode = 1
      if (hwtp1 .eq. 'FM') kode = 1
      if (hwtp1 .eq. 'SPUR') kode = 1
      if (hwtp1 .eq. 'BW') kode = 1
      if (hwtp1 .eq. 'P') kode = 1
      if (hwtp1 .eq. 'TOLL') kode = 1
      if (kode.eq.0) then
          goto 7
     else
     dummy4 = setbkcolor(int4(0))
     call settextposition(8,int2(col),curpos)
     call outtext (hwtp1)
     end if
     col=col+9
     dummy4=setbkcolor(int4(4))
     call settextposition(8,int2(col),curpos)
     call outtext('    ')
     call settextposition(8,int2(col),curpos)
     read(*,3) hwno1
     dummy4=setbkcolor(int4(0))
     call settextposition(8,int2(col),curpos)
     call outtext(hwno1)

6     dummy4=setbkcolor(int4(0))
     col=col+18
     dummy4 = setbkcolor(int4(4))
     call settextposition(8,int2(col),curpos)
     call outtext('    ')
     call settextposition(8,int2(col),curpos)
       kode = 0
       read(*,3) HWTP2
       if (hwtp2 .eq. 'IH') kode = 1
       if (hwtp2 .eq. 'US') kode = 1
       if (hwtp2 .eq. 'SH') kode = 1
       if (hwtp2 .eq. 'FM') kode = 1
       if (hwtp2 .eq. 'SPUR') kode = 1
       if (hwtp2 .eq. 'BW') kode = 1
       if (hwtp2 .eq. 'P') kode = 1
       if (hwtp2 .eq. 'TOLL') kode = 1
       if (kode.eq.0) then
           goto 6
      else
      dummy4 = setbkcolor(int4(0))
      call settextposition(8,int2(col),curpos)
      call outtext (hwtp2)
      end if
      col=col+10
      dummy4=setbkcolor(int4(4))
      call settextposition(8,int2(col),curpos)
      call outtext('    ')
      call settextposition(8,int2(col),curpos)
      read(*,3) hwno2
      dummy4=setbkcolor(int4(0))
      call settextposition(8,int2(col),curpos)
      call outtext(hwno2)


     i=1
     j=0

72    if (hwtp1.eq.nodhwt(i,1).and.hwno1.eq.nodhwn(i,1) ) then
      if( (hwtp2.eq.nodhwt(i,2).and.hwno2.eq.nodhwn(i,2)) .or.
     +      (hwtp2.eq.nodhwt(i,3).and.hwno2.eq.nodhwn(i,3)) ) then
        j=j+1
        in(j) = node(i)
```

```
      endif
     endif
     i=i+1
     if(i.le.nn) goto 72

     i=1
73    if (hwtp1.eq.nodhwt(i,2).and.hwno1.eq.nodhwn(i,2) ) then
      if( (hwtp2.eq.nodhwt(i,1).and.hwno2.eq.nodhwn(i,1)) .or.
    +      (hwtp2.eq.nodhwt(i,3).and.hwno2.eq.nodhwn(i,3)) ) then
       j=j+1
       in(j) = node(i)
      endif
     endif
     i=i+1
     if(i.le.nn) goto 73

     i=1
74    if (hwtp1.eq.nodhwt(i,3).and.hwno1.eq.nodhwn(i,3) ) then
      if( (hwtp2.eq.nodhwt(i,2).and.hwno2.eq.nodhwn(i,2)) .or.
    +      (hwtp2.eq.nodhwt(i,1).and.hwno2.eq.nodhwn(i,1)) ) then
       j=j+1
       in(j) = node(i)
      endif
     endif
     i=i+1
     if(i.le.nn) goto 74

     if(j.gt.1) then
                    dummy4 = setbkcolor(int4(1))
                          call clearscreen($gclearscreen)
                          write(str4,'(i4)') j
                          dummy4 = setbkcolor(int4(1))
                          call settextposition(int2(3),int2(14),curpos)
                          call outtext(' There are ')
                          call settextposition(int2(3),int2(26),curpos)
                          call outtext(str4)
                          call settextposition(int2(3),int2(31),curpos)
                          call outtext(' nodes which intersects the ')
                          call settextposition(int2(4),int2(14),curpos)
                          call outtext(' given highways.  These are ')
                          irow=5
                          do 999 k=1,j
        write(str4,'(i4)') in(k)
                                  irow=irow+1
        rewind(3)
1010      read(3,fmt=1026,end=1029) intnode,descnode
        if(intnode.ne.in(k)) goto 1010
1029      continue
        REWIND(3)
1026      format(i5,69x,a18)
                          call settextposition(int2(irow),int2(14),curpos)
                                  call outtext(str4)
                                  call settextposition(int2(irow),int2(19),curpos)
                          call outtext(descnode)
999                 continue

       irow=irow+2
1001    call settextposition(int2(irow),int2(19),curpos)
                          call outtext('Please Enter your Choice number ')

                          dummy4=setbkcolor(int4(4))
                          call settextposition(int2(irow),int2(52),curpos)
                          call outtext('    ')
                          dummy4=setbkcolor(int4(1))
                          call settextposition(int2(irow),int2(52),curpos)
       read(*,*) stnode

       kode=0
       do 1002 k=1,j
```

```
                                    if (stnode.eq.in(k)) then
                                            kode = 1
                                          endif
1002     continue
1003     if (kode.eq.0) goto 1001

c        write(*,*)'There are',j,'nodes which intersects the given
c     +   highways'
c        write(*,*) 'Enter the correct node no of these nodes'
c        write(*,*) (in(k),k=1,j)
c        read(*,*) stnode
      endif
      if(j.eq.0) then
       dummy4 = setbkcolor(int4(0))
       call clearscreen($gclearscreen)
       dummy4 = setbkcolor(int4(4))
       call settextposition(int2(4),int2(16),curpos)
       call outtext('                                  ')
       call settextposition(int2(5),int2(16),curpos)
       call outtext('*******************************************   ')
       call settextposition(int2(6),int2(16),curpos)
       call outtext('                ERROR                  ')
       call settextposition(int2(5),int2(16),curpos)
       call outtext('*******************************************   ')

       call settextposition(int2(7),int2(16),curpos)
       call outtext(' NO NODE WHICH INTERSECTS THESE HIGHWAYS        ')
       call settextposition(int2(8),int2(16),curpos)
       call outtext('                                  ')
       call settextposition(int2(9),int2(16),curpos)
       call outtext(' PRESS ANY KEY TO REINPUT THE NODE DATA        ')
       call settextposition(int2(10),int2(16),curpos)
       call outtext('*******************************************   ')
       read(*,*)
       goto 101
c        write(*,*)'No node which intersects the highways'
      endif

      if(j.eq.1) then
       dummy4=setbkcolor(int4(1))
       call settextposition(int2(10),int2(16),curpos)
                    call outtext(' The node which intersects the given highways ')
                    call settextposition(int2(11),int2(16),curpos)
                    call outtext(' is :                               ')
                    write(str4,'(i4)') in(j)
                    call settextposition(int2(11),int2(22),curpos)
                    call outtext(str4)
c        write(*,*) 'The node intersects the given highways is',in(j)
       stnode=in(j)
       endif

      read(*,*)

c#####################################################################

      ans = ' '
c      stnode = 0
c      do while (stnode.lt.1.or.stnode.gt.233)
c      dummy4 = setbkcolor(int4(0))
c      call clearscreen ($gclearscreen)
c      dummy4 = setbkcolor(int4(4))
c      call settextposition(INT2(2),INT2(15),curpos)
c      call outtext(' Please input STARTING NODE number: ')
c      dummy4 = setbkcolor(int4(9))
c      call settextposition(INT2(2),INT2(52),curpos)
c      call outtext('   ')
c      dummy4 = setbkcolor(int4(0))
c      call settextposition(INT2(2),INT2(52),curpos)
c      read(*,*) stnode
```

```
c      if (stnode.lt.1.or.stnode.gt.233) then
c        dummy4 = setbkcolor(int4(4))
c        call settextposition(INT2(18),INT2(16),curpos)
c        call outtext(' **** Illegal entry -- must be greater than 0 ')
c        call settextposition(INT2(19),INT2(16),curpos)
c        call outtext('      and less than 234                     ')
c        call settextposition(INT2(21),INT2(29),curpos)
c        call outtext(' Press ENTER to continue ')
c        read(*,*)
c      endif
c     end do

c 6    write(*,*) '    Enter STARTING NODE number'
c     READ(5,*) STNODE
c     If (stnode.lt.1.or.stnode.gt.233) then
c     write (*,*) '**** Illegal entry -- must be greater than 0 and '
c     write (*,*) '     less than 234'
c     goto 6
c     end if


C######################################################################
201   j=1
     dummy4=setbkcolor(int4(0))
     call clearscreen($gclearscreen)
     dummy4=setbkcolor(int4(1))
     call settextposition(3,int2(18),curpos)
     call outtext(' ENTER INTERSECTING HIGHWAYS FOR END POINT    ')
     call settextposition(6,int2(18),curpos)
     call outtext(' HW1_TYPE  HW1_NO           HW2_TYPE   HW2_NO')

777     dummy4=setbkcolor(int4(0))
     col=j+20
     dummy4 = setbkcolor(int4(4))
     call settextposition(8,int2(col),curpos)
     call outtext('    ')
     call settextposition(8,int2(col),curpos)
       kode = 0
       read(*,3) HWTP1
       if (hwtp1 .eq. 'IH') kode = 1
       if (hwtp1 .eq. 'US') kode = 1
       if (hwtp1 .eq. 'SH') kode = 1
       if (hwtp1 .eq. 'FM') kode = 1
       if (hwtp1 .eq. 'SPUR') kode = 1
       if (hwtp1 .eq. 'BW') kode = 1
       if (hwtp1 .eq. 'P') kode = 1
       if (hwtp1 .eq. 'TOLL') kode = 1
       if (kode.eq.0) then
           goto 777
      else
      dummy4 = setbkcolor(int4(0))
      call settextposition(8,int2(col),curpos)
      call outtext (hwtp1)
      end if
      col=col+9
      dummy4=setbkcolor(int4(4))
      call settextposition(8,int2(col),curpos)
      call outtext('    ')
      call settextposition(8,int2(col),curpos)
      read(*,3) hwno1
      dummy4=setbkcolor(int4(0))
      call settextposition(8,int2(col),curpos)
      call outtext(hwno1)

666     dummy4=setbkcolor(int4(0))
     col=col+18
     dummy4 = setbkcolor(int4(4))
     call settextposition(8,int2(col),curpos)
     call outtext('    ')
```

```
call settextposition(8,int2(col),curpos)
  kode = 0
  read(*,3) HWTP2
  if (hwtp2 .eq. 'IH') kode = 1
  if (hwtp2 .eq. 'US') kode = 1
  if (hwtp2 .eq. 'SH') kode = 1
  if (hwtp2 .eq. 'FM') kode = 1
  if (hwtp2 .eq. 'SPUR') kode = 1
  if (hwtp2 .eq. 'BW') kode = 1
  if (hwtp2 .eq. 'P') kode = 1
  if (hwtp2 .eq. 'TOLL') kode = 1
  if (kode.eq.0) then
      goto 666
 else
 dummy4 = setbkcolor(int4(0))
 call settextposition(8,int2(col),curpos)
 call outtext (hwtp2)
 end if
 col=col+10
 dummy4=setbkcolor(int4(4))
 call settextposition(8,int2(col),curpos)
 call outtext('    ')
 call settextposition(8,int2(col),curpos)
 read(*,3) hwno2
 dummy4=setbkcolor(int4(0))
 call settextposition(8,int2(col),curpos)
 call outtext(hwno2)



i=1
j=0

722   if (hwtp1.eq.nodhwt(i,1).and.hwno1.eq.nodhwn(i,1) ) then
     if( (hwtp2.eq.nodhwt(i,2).and.hwno2.eq.nodhwn(i,2)) .or.
+         (hwtp2.eq.nodhwt(i,3).and.hwno2.eq.nodhwn(i,3)) ) then
       j=j+1
       in(j) = node(i)
     endif
    endif
    i=i+1
    if(i.le.nn) goto 722

i=1
733   if (hwtp1.eq.nodhwt(i,2).and.hwno1.eq.nodhwn(i,2) ) then
     if( (hwtp2.eq.nodhwt(i,1).and.hwno2.eq.nodhwn(i,1)) .or.
+         (hwtp2.eq.nodhwt(i,3).and.hwno2.eq.nodhwn(i,3)) ) then
       j=j+1
       in(j) = node(i)
     endif
    endif
    i=i+1
    if(i.le.nn) goto 733

i=1
744   if (hwtp1.eq.nodhwt(i,3).and.hwno1.eq.nodhwn(i,3) ) then
     if( (hwtp2.eq.nodhwt(i,2).and.hwno2.eq.nodhwn(i,2)) .or.
+         (hwtp2.eq.nodhwt(i,1).and.hwno2.eq.nodhwn(i,1)) ) then
       j=j+1
       in(j) = node(i)
     endif
    endif
    i=i+1
    if(i.le.nn) goto 744

if(j.gt.1) then
                dummy4 = setbkcolor(int4(1))
                        call clearscreen($gclearscreen)
                        write(str4,'(i4)') j
```

```
                        dummy4=setbkcolor(int4(1))
                        call settextposition(int2(3),int2(14),curpos)
                        call outtext(' There are ')
                        call settextposition(int2(3),int2(26),curpos)
                        call outtext(str4)
                        call settextposition(int2(3),int2(31),curpos)
                        call outtext(' nodes which intersects the ')
                        call settextposition(int2(4),int2(14),curpos)
                        call outtext(' given highways.  These are ')
                        irow=5
                        do 995 k=1,j
      write(str4,'(i4)') in(k)
                             irow=irow+1
      rewind(3)
1030    read(3,fmt=1026,end=1039) intnode,descnode
      if(intnode.ne.in(k)) goto 1030
1039    continue
      REWIND(3)
                        call settextposition(int2(irow),int2(14),curpos)
                             call outtext(str4)
                             call settextposition(int2(irow),int2(19),curpos)
                             call outtext(descnode)
995              continue

     irow=irow+2
1004   call settextposition(int2(irow),int2(19),curpos)
                        call outtext('Please Enter your Choice number ')
                        dummy4=setbkcolor(int4(4))
                        call settextposition(int2(irow),int2(52),curpos)
                        call outtext('    ')
                        dummy4=setbkcolor(int4(1))
                        call settextposition(int2(irow),int2(52),curpos)
     read(*,*) endnode

     kode=0
     do 1005 k=1,j
                         if (endnode.eq.in(k)) then
                              kode=1
                             endif
1005   continue
1006   if (kode.eq.0) goto 1004

c      write(*,*)'There are',j,'nodes which intersects the given
c    +   highways'
c      write(*,*) 'Enter the correct node no of these nodes'
c      write(*,*) (in(k),k=1,j)
c      read(*,*) endnode
    endif

    if(j.eq.0) then
      dummy4 = setbkcolor(int4(0))
      call clearscreen($gclearscreen)
      dummy4 = setbkcolor(int4(4))
      call settextposition(int2(4),int2(16),curpos)
      call outtext('                                  ')
      call settextposition(int2(5),int2(16),curpos)
      call outtext('*******************************************  ')
      call settextposition(int2(6),int2(16),curpos)
      call outtext('                ERROR                  ')
      call settextposition(int2(5),int2(16),curpos)
      call outtext('*******************************************  ')

      call settextposition(int2(7),int2(16),curpos)
      call outtext(' NO NODE WHICH INTERSECTS THESE HIGHWAYS       ')
      call settextposition(int2(8),int2(16),curpos)
      call outtext('                                  ')
      call settextposition(int2(9),int2(16),curpos)
      call outtext(' PRESS ANY KEY TO REINPUT THE NODE DATA        ')
      call settextposition(int2(10),int2(16),curpos)
```

```
      call outtext('*******************************************   ')
      read(*,*)
      goto 201
c       write(*,*)'No node which intersects the highways'
     endif


     if(j.eq.1) then
       call settextposition(int2(10),int2(16),curpos)
                         call outtext('The node which intersects the given highways ')
                         call settextposition(int2(11),int2(16),curpos)
                         call outtext('is : ')
                         write(str4,'(i4)') in(j)
                         call settextposition(int2(11),int2(22),curpos)
                         call outtext(str4)
c       write(*,*) 'The node intersects the given highways is',in(j)
      endnode=in(j)
      endif
     read(*,*)

c######################################################################


c     endnode = 0
c     do while (endnode.lt.1.or. endnode.gt.233)
c     dummy4 = setbkcolor(int4(0))
c     call clearscreen ($gclearscreen)
c     dummy4 = setbkcolor(int4(4))
c     call settextposition(INT2(2),INT2(15),curpos)
c     call outtext(' Please input ENDING NODE number: ')
c     dummy4 = setbkcolor(int4(9))
c     call settextposition(INT2(2),INT2(51),curpos)
c     call outtext('   ')
c     dummy4 = setbkcolor(int4(0))
c     call settextposition(INT2(2),INT2(51),curpos)
c     read(*,*) endnode

c      if (endnode.lt.1.or. endnode.gt.233) then
c        dummy4 = setbkcolor(int4(4))
c        call settextposition(INT2(18),INT2(16),curpos)
c        call outtext(' **** Illegal entry -- must be greater than 0 ')
c        call settextposition(INT2(19),INT2(16),curpos)
c        call outtext('      and less than 234                    ')
c        call settextposition(INT2(21),INT2(29),curpos)
c        call outtext(' Press ENTER to continue ')
c        read(*,*)
c      endif
c     end do



c 7    write(*,*) '    Enter ENDING NODE number'
c     READ(5,*) ENDNODE
c    if (ENDNODE.lt.1.or.ENDNODE.gt.233) then
c     write (*,*) '**** Illegal entry -- must be greater than 0 and '
c     write (*,*) '     less than 234'
c     goto 7
c     end if
C
C
C

     J=1
     dummy4 = setbkcolor(int4(0))
     call clearscreen ($gclearscreen)
     dummy4 = setbkcolor(int4(1))
     call settextposition(2,int2(26),curpos)
     call outtext(' ENTER ROUTE INFORMATION ')
     call settextposition(4,int2(18),curpos)
```

```
      call outtext(' SEGMENT    HWY_TYPE    HWY_NO    DIRECTION ')
8      dummy4 = setbkcolor(int4(0))
      row = j+5

      call settextposition(row,INT2(22),curpos)
      write(str1,'(I1)') j
      call outtext( str1)
      dummy4 = setbkcolor(int4(4))
      call settextposition(row,int2(32),curpos)
      call outtext('    ')
      call settextposition(row,int2(32),curpos)
c      call outtext(' Enter HIGHWAY TYPE (I,US,SH,FM,..) OR END')
        kode = 0
        read(*,3) HWTYP(J)
 3      format (a4)
        IF (HWTYP(J) .EQ. 'END') GO TO 25
        if (hwtyp(j) .eq. 'IH') kode = 1
        if (hwtyp(j) .eq. 'US') kode = 1
        if (hwtyp(j) .eq. 'SH') kode = 1
        if (hwtyp(j) .eq. 'FM') kode = 1
        if (hwtyp(j) .eq. 'SPUR') kode = 1
        if (hwtyp(j) .eq. 'BW') kode = 1
        if (hwtyp(j) .eq. 'P') kode = 1
        if (hwtyp(j) .eq. 'TOLL') kode = 1
        if (kode.eq.0) then
c             dummy4 = setbkcolor(int4(4))
c             call settextposition(INT2(16),INT2(15),curpos)
c             call outtext('    ****** illegal entry *****    ')
c             call settextposition(INT2(20),INT2(22),curpos)
c             call outtext('Press ENTER to continue ')
c             read(*,*)
            goto 8
        else
        dummy4 = setbkcolor(int4(0))
        call settextposition(row,int2(32),curpos)
        call outtext (hwtyp(j))
        end if
        dummy4 = setbkcolor(int4(4))
        call settextposition (row, int2(43),curpos)
        call outtext('    ')
        call settextposition(row, int2(43),curpos)

c      call outtext(' Enter HIGHWAY NUMBER ')
      READ(*,3) HWNO(J)
      dummy4 = setbkcolor(int4(0))
      call settextposition (row,int2(43),curpos)
      call outtext(hwno(j))
9      continue
        dummy4 = setbkcolor(int4(4))
        call settextposition(row,INT2(55),curpos)
c          call outtext(' Enter direction to follow (E,W,N OR S)' )
        call outtext(' ')
        call settextposition(row, int2(55),curpos)
        kode = 0
        READ(*,2) DIR(J)
 2       format(a1)
        if(dir(j).eq.'E') kode=1
        if(dir(j).eq.'W') kode=1
        if(dir(j).eq.'N') kode=1
        if(dir(j).eq.'S') kode=1
        if (kode.eq.0) then
c             dummy4 = setbkcolor(int4(4))
c             call settextposition(INT2(16),INT2(15),curpos)
c             call outtext('    ****** illegal entry *****    ')
c             call settextposition(INT2(20),INT2(22),curpos)
c             call outtext('Press ENTER to continue ')
c             read(*,*)
            goto 9
        else
```

```
          dummy4 = setbkcolor(int4(0))
          call settextposition(row,int2(55),curpos)
          call outtext(dir(j))
         end if

       J=J+1
        GOTO 8
c
25    CONTINUE
      dummy4 = setbkcolor(int4(0))
      call clearscreen($gclearscreen)
c
      NSEG = J -1



c ****
      dummy4 = setbkcolor(int4(1))
      call clearscreen($gclearscreen)
      write(str4,'(I4)') nseg
c     WRITE(*,30) nseg
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(3),int2(14),curpos)
      call outtext(' You have selected to travel over')
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(3),int2(48),curpos)
      call outtext(str4)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(3),int2(53),curpos)
      call outtext(' segment(s) ')
      write(str4,'(I4)') stnode
      call settextposition(int2(5),int2(14),curpos)
      call outtext(' begining at node ')
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(5),int2(33),curpos)
      call outtext(str4)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(5),int2(40),curpos)
      call outtext('that intersects highways ' )
      write(str4,'(a4)')nodhwt(stnode,1)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(6),int2(15),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwn(stnode,1)
      call settextposition(int2(6),int2(19),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwt(stnode,2)
      call settextposition(int2(6),int2(28),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwn(stnode,2)
      call settextposition(int2(6),int2(32),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwt(stnode,3)
      call settextposition(int2(6),int2(39),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwn(stnode,3)
      call settextposition(int2(6),int2(43),curpos)
      call outtext(str4)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(8),int2(14),curpos)
      call outtext(' ending  at  node ')
      write(str4,'(i4)')endnode
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(8),int2(33),curpos)
      call outtext(str4)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(8),int2(40),curpos)
      call outtext('that intersects highways    ')
      write(str4,'(a4)')nodhwt(endnode,1)
```

```
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(9),int2(15),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwn(endnode,1)
      call settextposition(int2(9),int2(19),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwt(endnode,2)
      call settextposition(int2(9),int2(28),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwn(endnode,2)
      call settextposition(int2(9),int2(32),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwt(endnode,3)
      call settextposition(int2(9),int2(39),curpos)
      call outtext(str4)
      write(str4,'(a4)')nodhwn(endnode,3)
      call settextposition(int2(9),int2(43),curpos)
      call outtext(str4)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(11),int2(14),curpos)
      call outtext(' following: ')
      row=12

      do 35 i=1,nseg
        dummy4 = setbkcolor(int4(1))
        write(str4,'(a4)') hwtyp(i)
        call settextposition(int2(row),int2(15),curpos)
        call outtext(str4)
        write(str4,'(a4)') hwno(i)
        call settextposition(int2(row),int2(21),curpos)
        call outtext(str4)
        write(str1,'(a1)') dir(i)
        call settextposition(int2(row),int2(28),curpos)
        call outtext(str1)
        row=row+1
 35   continue
      dummy4 = setbkcolor(int4(0))

c=========================================
c     WRITE(1,30) nseg
c 30   format (5x,'You have selected to travel over ',i3,' segments')
c      write(*,31) stnode
c      write(1,31) stnode
c 31   format (5x,'beginning at node ',i4,' that intersects highways')
c      write(*,32) nodhwt(stnode,1),nodhwn(stnode,1),nodhwt(stnode,2),
c     &    nodhwn(stnode,2),nodhwt(stnode,3),nodhwn(stnode,3)
c      write(1,32) nodhwt(stnode,1),nodhwn(stnode,1),nodhwt(stnode,2),
c     &    nodhwn(stnode,2),nodhwt(stnode,3),nodhwn(stnode,3)
c 32   format (10x,3(a4,1x,A4,3x))
c      write(*,33) endnode
c      write(1,33) endnode
c 33   format (5x,'ending at node ',i4,' that intersects highways')
c      write(*,32) nodhwt(endnode,1),nodhwn(endnode,1),nodhwt(endnode,2),
c     &    nodhwn(endnode,2),nodhwt(endnode,3),nodhwn(endnode,3)
c      write(*,34)
c      write(1,32) nodhwt(endnode,1),nodhwn(endnode,1),nodhwt(endnode,2),
c     &    nodhwn(endnode,2),nodhwt(endnode,3),nodhwn(endnode,3)
c      write(1,34)
c 34   format(/,5x,'following:',/)
c      do 35 i=1,nseg
c      write(*,36) hwtyp(i),hwno(i),dir(i)
c      write(1,36) hwtyp(i),hwno(i),dir(i)
c 36   format (10x,a4,1x,A4, ' direction ',a1)
c 35   continue
c=============================================================

      do while (ans.ne.'Y'.and. ans.ne.'N')
      dummy4 = setbkcolor(int4(0))
      call settextposition(int2(24),int2(20),curpos)
```

```
      call outtext (' Is this correct (Y/N) ? ')
      dummy4 = setbkcolor(int4(4))
      call settextposition(int2(24),int2(47),curpos)
      call outtext(' ')
      call settextposition(int2(24),int2(47),curpos)
      read(*,57) ans
57    FORMAT (A1)
      end do
C     pause '                          hit <enter> to continue'
      dummy4 = setbkcolor(int4(0))
      if (ans.eq.'N') goto 54

      dummy4 = setbkcolor(int4(0))
      call clearscreen($gclearscreen)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(12),int2(31),curpos)
      call outtext(' Please Wait .... ')

C
C
c        read link data
c
      k = 1
 40   read(2,fmt=41,end=42) link(k),nc(k,1),nc(k,2),xl(k),l,
     &      (lhwt(k,i),lhwn(k,i),i=1,l)
      ndef(k) = l
      xl(k) = xl(k)/5280.
C     write(*,41) link(k),nc(k,1),nc(k,2),xl(k),ndef(k),
C    &      (lhwt(k,i),lhwn(k,i),i=1,ndef(k))
      k = k+1
      goto 40
 42   continue
 41   format(3(i5),f15.3,i3,3(1x,a4,1x,a4))
      nl = k -1
c
c      read bridge attributes of links
c
      do 45 i=1,1100
      read(4,50) jj, nb(i),(ibr(i,j),j=1,nb(i))
C     write(*,50) jj, nb(i),(ibr(i,j),j=1,nb(i))
45    continue
 50   format(i5,i7,5x,50(i10))
      K = 0
      DO 55 I=1,1100
      DO 55 J=1,nb(i)
      k = k+1
      idb(K) = ibr(i,j)
 55   continue
      nbatt = k
c     WRITE(1,*) 'NBATT=',NBATT
c     DO 56 I=1,NBATT
c 56  write(1,*) idb(i)


      I=1
283     READ(11,282,END=285) LK1,LK2,NOSTR,(ISTR(I,J),J=1,NOSTR)
282     FORMAT(I4,I4,I2,25(I10))
      DO 62 J=1,NOSTR
       k = k+1
       idb(K) = ISTR(i,j)
 62    continue
C      NB2=NB2+NOSTR
      I=I+1
      GOTO 283
285     REWIND 11

      nbatt = k

c
```

```
c     sort
c
      do 60 i=1,nbatt
      do 70  j=i,nbatt
      if (idb(j).lt. idb(i)) then
C        write(*,*)'j',j
       itemp = idb(i)
       idb(i)= idb(j)
       idb(j) = itemp
      end if
 70  continue
 60  continue
c     WRITE(1,*) 'BEFORE ELIMINATION'
c     do 75 i=1,nbatt
c     write(1,*) idb(i)
c 75   continue

      IF(NBATT.EQ.0) GOTO 83
      k = 1
      kk = 2
 77   if (idb(kk) .gt.idb(k)) then
      k = k + 1
      idb(k) = idb(kk)
      else
      kk = kk+1
      end if
      if (kk.le.nbatt) goto 77
      nb2 = k

c     WRITE(1,*) 'AFTER ELIMINATION'
c     do 76 i=1,nb2
c     write(1,*) idb(i)
c 76   continue


c     WRITE(*,*) ' HARRIS COUNTY NETWORK INFORMATION'
c     WRITE(*,*) ' *********************************'
c     WRITE(*,*) ' '
c     write(*,*) '     Number of nodes ',nn
c     write(*,*) '     Number of links ',nl
c     write(*,*) '     Number of total bridge attributes',nbatt
c     WRITE(*,*) '       Number of total bridges',nb2

83   dummy4 = setbkcolor(int4(0))
      call clearscreen($gclearscreen)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(4),int2(23),curpos)
      call outtext('                              ')
      call settextposition(int2(5),int2(23),curpos)
      call outtext(' HOUSTON DISTRICT NETWORK INFO      ')
      call settextposition(int2(6),int2(23),curpos)
      call outtext(' ****************************        ')
      call settextposition(int2(7),int2(23),curpos)
      call outtext('                              ')
      call settextposition(int2(8),int2(23),curpos)
      call outtext('    Number of nodes                ')
      write(str4,'(i4)') nn
      call settextposition(int2(8),int2(47),curpos)
      call outtext(str4)
      call settextposition(int2(9),int2(23),curpos)
      call outtext('    Number of links                ')
      write(str4,'(i4)') nl
      call settextposition(int2(9),int2(47),curpos)
      call outtext(str4)
      call settextposition(int2(10),int2(23),curpos)
      call outtext('    Total Number of bridges         ')
      write(str4,'(i4)') nb2
      call settextposition(int2(10),int2(54),curpos)
      call outtext(str4)
```

```
      call settextposition(int2(11),int2(23),curpos)
      call outtext('                                  ')
      dummy4 = setbkcolor(int4(25))
      call settextposition(int2(23),int2(29),curpos)
      call outtext(' Press ENTER to continue ')
      dummy4 = setbkcolor(int4(0))



      WRITE(1,*) ' HOUSTON DISTRICT NETWORK INFORMATION '
      WRITE(1,*) ' **********************************  '
      WRITE(1,*) ' '
      write(1,*) '     Number of nodes ',nn
      write(1,*) '     Number of links ',nl
C     write(1,*) '     Number of total bridge attributes',nbatt
      WRITE(1,*) '     Total number of bridges ',nb2
      pause ' '
C     pause '                              hit <ENTER> to continue'
C
C     DETERMINE NUMBER OF LINKS CONNECTING TO NODE
C
      DO 90 I = 1, NL
      DO 95 J=1,2
      NOD = NC(I,J)
      NLNOD(NOD) = NLNOD(NOD)+1
      K = NLNOD(NOD)
      LNUM(NOD,K) = I
C     WRITE(*,*) I
 95   CONTINUE
 90   CONTINUE
C     OPEN (7,FILE='LLL.DAT',STATUS='NEW')
C     DO 100 I=1,NN
C     WRITE(7,*)I, NLNOD(I),(LNUM(I,J),J=1,NLNOD(I))
C100  CONTINUE
C
C     BEGIN THE ROUTE SELECTION OF THE ROUTE
C
      ISEG=1
      K = 0

      NUMBER = NLNOD(STNODE)
      NOD = STNODE
C ************

 150  KODE = 0
      kode2=0
      NUMBER = NLNOD(NOD)
      DO 105 I=1, NUMBER
       IF(KODE.NE.0) GOTO 105
       LN = LNUM(NOD,I)
                        IF (K.GT.0) THEN
                          IF (LN.EQ.LTRAVEL(K)) GOTO 105
                        END IF
C       WRITE(*,*) 'LN  ISEG', LN,ISEG
C+ + + + + + + + + + + +
       n = ndef(ln)
       iflag = 0
       do j = 1, n
       IF (HWTYP(ISEG).eq.LHWT(LN,j)) then
         IF (HWNO(ISEG).eq.LHWN(LN,j)) iflag = 1
       end if
       end do
       if (iflag.eq.0) goto 105
C      at this point the link  ln in on the path of travel
c      defined on the current segment (iseg)

       IF (NC(LN,1).EQ.NOD) THEN
         NODE1 = NC(LN,1)
         NODE2 = NC(LN,2)
```

```
      END IF
      IF (NC(LN,2).EQ.NOD) THEN
        NODE1 = NC(LN,2)
        NODE2 = NC(LN,1)
      END IF
        XXL = SQRT((X(NODE2)-X(NODE1))**2+(Y(NODE2)-Y(NODE1))**2)
        CO = (X(NODE2)-X(NODE1))/XXL
        SI = (Y(NODE2)-Y(NODE1))/XXL
        IF(DIR(ISEG).EQ.'E') THEN
          IF(CO.GT.0.) KODE=1
        END IF
        IF (DIR(ISEG).EQ.'W') THEN
          IF(CO.LT.0.) KODE=1
        END IF
        IF (DIR(ISEG).EQ.'N') THEN
          IF(SI.GT.0.) KODE=1
        END IF
        IF (DIR(ISEG).EQ.'S') THEN
          IF(SI.LT.0.) KODE=1
        END IF
c
c       if kode is 1 the link ln also has the same direction
c       for the current seg, otherwise the correct direction was
c       not found
C
        kode2 = 1

105   CONTINUE
        IF(KODE.EQ.0 .and.kode2.ne.0 ) THEN
          IF(HWTYP(ISEG+1).EQ.HWTYP(ISEG).AND.
     &       HWNO(ISEG+1).EQ.HWNO(ISEG)) then
                      ISEG = ISEG+1
C                      write(*,*) '***',hwtyp(iseg),hwno(iseg),dir(iseg),LTRAVEL(K)
            GOTO 150
          end if
            END IF

     if (kode.eq.1) then
          K = K+1
          LTRAVEL(K) = LN
          NTRAVEL(K) = NOD
          NOD = NODE2
C           WRITE(*,*) 'LTRAVEL',LTRAVEL(K)
C           WRITE(*,*) 'NTRAVEL',NTRAVEL(K)
C           WRITE(*,*) 'NEXT NODE',NOD
                                NOD = NODE2
        IF (NOD.EQ.ENDNODE) GOTO 200
C
C            CHECK TO SEE IF YOU NEED TO SWITCH SEGMENTS
C
           NUMBER = NLNOD(NOD)
           DO 103 I = 1,NUMBER
             LN = LNUM(NOD,I)
             IF(LN.EQ.LTRAVEL(K)) GOTO 103
             IF(NOD.EQ.NC(LN,1)) THEN
             N1 = NC(LN,1)
             N2 = NC(LN,2)
             ELSE
             N1 = NC(LN,2)
             N2 = NC(LN,1)
             END IF
             XXL = SQRT( (X(N2) - X(N1))**2 + (Y(N2)-Y(N1))**2)
             CO = (X(N2)-X(N1))/XXL
             SI = (Y(N2)-Y(N1))/XXL
C              WRITE(*,*)'CO,SI',CO,SI
C +++++++++++++++
             n = ndef (ln)
             iflag=0
             do j=1,n
```

```
          IF (HWTYP(ISEG+1).EQ.LHWT(LN,j)) THEN
           IF (HWNO(ISEG+1).EQ.LHWN(LN,j)) iflag=1
          end if
         end do
         if (iflag.eq.1) then
            IF(HWTYP(ISEG+1).EQ.HWTYP(ISEG).AND.
&              HWNO(ISEG+1).EQ.HWNO(ISEG)) THEN
             IF (DIR(ISEG+1).EQ.'N') THEN
               IF(ABS(SI).GT.ABS(CO).AND.SI.GT.0.) KODE=2
             END IF
             IF (DIR(ISEG+1).EQ.'S') THEN
               IF(ABS(SI).GT.ABS(CO).AND.SI.LT.0.) KODE=2
             END IF
             IF (DIR(ISEG+1).EQ.'E') THEN
               IF(ABS(SI).LT.ABS(CO).AND.CO.GT.0.) KODE=2
             END IF
             IF (DIR(ISEG+1).EQ.'W') THEN
               IF(ABS(SI).LT.ABS(CO).AND.CO.LT.0.) KODE=2
             END IF
            ELSE
             KODE = 2
            END IF
         END IF
 103      CONTINUE
   end if
   IF (KODE.EQ.1) GOTO 150
   IF (KODE.EQ.2) THEN
    ISEG = ISEG+1
C                    write(*,*) hwtyp(iseg),hwno(iseg),dir(iseg),LTRAVEL(K)
    GOTO 150
   END IF
   IF(KODE.EQ.0.AND.KODE2.EQ.0) THEN
C     WRITE(*,*) '**********************************************'
C          WRITE(*,*) '                ERROR'
C     WRITE(*,*) '**********************************************'
    dummy4 = setbkcolor(int4(0))
    call clearscreen($gclearscreen)
    dummy4 = setbkcolor(int4(4))
    call settextposition(int2(4),int2(16),curpos)
    call outtext('                                            ')
    call settextposition(int2(5),int2(16),curpos)
    call outtext('********************************************  ')
    call settextposition(int2(6),int2(16),curpos)
    call outtext('                ERROR                  ')
    call settextposition(int2(5),int2(16),curpos)
    call outtext('********************************************  ')

    WRITE(1,*) '**********************************************'
         WRITE(1,*) '                ERROR'
    WRITE(1,*) '**********************************************'


C     WRITE(*,*) 'TRAVEL ROUTE FAILED TO CONVERGE TO ITS POINT OF'
C     WRITE(*,*) 'DESTINY, CHECK YOUR INPUT FOR THE ROUTE '
C     WRITE(*,*) 'DEFINITION'
C          WRITE(*,*) '**********************************************'
    call settextposition(int2(7),int2(16),curpos)
    call outtext(' TRAVEL ROUTE FAILED TO CONVERGE TO ITS POINT   ')
    call settextposition(int2(8),int2(16),curpos)
    call outtext(' OF DESTINY, CHECK YOUR INPUT FOR THE ROUTE     ')
    call settextposition(int2(9),int2(16),curpos)
    call outtext(' DEFINITION                              ')
    call settextposition(int2(10),int2(16),curpos)
    call outtext('********************************************  ')
    call settextposition(int2(12),int2(16),curpos)
    call outtext('            Program is Exiting           ')
    call settextposition(int2(13),int2(16),curpos)
    call outtext('                                            ')
```

```
      dummy4 = setbkcolor(int4(0))
      call settextposition(int2(20),int2(16),curpos)
      call outtext('                                  ')

C       WRITE(*,*)
      WRITE(1,*) 'TRAVEL ROUTE FAILED TO CONVERGE TO ITS POINT OF'
      WRITE(1,*) 'DESTINY, CHECK YOUR INPUT FOR THE ROUTE '
      WRITE(1,*) 'DEFINITION'
      WRITE(1,*) '*********************************************'
      WRITE(1,*)
c       PAUSE ' '
      GOTO 5001
    END IF

200  CONTINUE
    NLINKS = K
    NNODES = K+1
    NTRAVEL(NNODES) = NOD



c     WRITE(*,*) '     LINKS TO BE TRAVELLED'
c     WRITE(*,*) '     *********************'
c     WRITE(*,180) (LTRAVEL(I),I=1,NLINKS)
    dummy4 = setbkcolor(int4(0))
    call clearscreen($gclearscreen)
    dummy4 = setbkcolor(int4(1))
    call settextposition(int2(4),int2(16),curpos)
    call outtext('          LINKS TO BE TRAVELLED             ')
    call settextposition(int2(5),int2(16),curpos)
    call outtext('          ********************               ')
    call settextposition(int2(6),int2(16),curpos)
    call outtext('                                  ')
    row=2
    col=16
    j=0
    call settextposition(int2(row),int2(16),curpos)
    call outtext('                                  ')
    do i=1,nlinks
      write(str6,'(I6)') ltravel(i)
      call settextposition(int2(row),int2(col),curpos)
      call outtext(str6)
      col = col + 6
      j=j+1
      if(j.eq.8) then
        row = row +1
        col = 16
        j=0
        call settextposition(int2(row),int2(16),curpos)
    call outtext('                                  ')
      endif
    enddo

    WRITE(1,*) '     LINKS TO BE TRAVELLED'
    WRITE(1,*) '     *********************'
    WRITE(1,180) (LTRAVEL(I),I=1,NLINKS)
 180  FORMAT(8(I6))
c     WRITE(*,*) ' '

                    XXL = 0.0
                    DO 288 I=1,NLINKS
                    XXL = XXL +XL(LTRAVEL(I))
 288  CONTINUE


c**********************************************************
c
c   loop to check for the two-way traffic links
```

```
c
c**********************************************************
      do 250 i=1,NLINKS
        ln=ltravel(i)
        node1=nc(ln,1)
        node2=nc(ln,2)
        if(node1.eq.ntravel(i+1) .and. node2.eq.ntravel(i)) then
221         read(6,225,end=245) lk1,lk2
          if (lk1 .eq. ltravel(i)) then
            ltravel(i) = lk2
            rewind 6
          else
            goto 221
          endif
        endif
245       rewind6
250   continue
225   format (i5,i5)

      WRITE(1,*) '     LINKS TO BE TRAVELLED WITH 2-WAY TRAFFIC'
      WRITE(1,*) '     ****************************************'
      WRITE(1,180) (LTRAVEL(I),I=1,NLINKS)
c 180  FORMAT(8(I6))
c     WRITE(*,*) ' '


c     WRITE(*,*) '     LINKS TO BE TRAVELLED'
c     WRITE(*,*) '     *********************'
c     WRITE(*,180) (LTRAVEL(I),I=1,NLINKS)

      row=row+2
      col=16
      j=0
      call settextposition(int2(row),int2(16),curpos)
      call outtext('                                        ')
      do i=1,nlinks
        write(str6,'(I6)') ltravel(i)
        call settextposition(int2(row),int2(col),curpos)
        call outtext(str6)
        col = col + 6
        j=j+1
        if(j.eq.8) then
          row = row +1
          col = 16
          j=0
          call settextposition(int2(row),int2(16),curpos)
      call outtext('                                        ')
        endif
      enddo


c**************************************************************


C     WRITE(*,*) '     NODES TO BE CROSSED'
C     WRITE(*,*) '     *******************'
C     WRITE(*,180) (NTRAVEL(I),I=1,NNODES)
      dummy4 = setbkcolor(int4(1))
      row=row+2
      call settextposition(int2(row),int2(16),curpos)
      call outtext('             NODES TO BE CROSSED              ')
      row=row+1
      call settextposition(int2(row),int2(16),curpos)
      call outtext('             *******************              ')
      row=row+1
      call settextposition(int2(row),int2(16),curpos)
      call outtext('                                        ')
      row=row+1
```

```
    col=16
    j=0
    call settextposition(int2(row),int2(16),curpos)
    call outtext('                                        ')
    do i=1,nnodes
      write(str6,'(I6)') ntravel(i)
      call settextposition(int2(row),int2(col),curpos)
      call outtext(str6)
      col = col + 6
      j=j+1
      if(j.eq.8) then
        row = row +1
        col = 16
        j=0
        call settextposition(int2(row),int2(16),curpos)
    call outtext('                                        ')
      endif
    enddo



    WRITE(1,*) ' '
    WRITE(1,*) '     NODES TO BE CROSSED'
    WRITE(1,*) '     *******************'
    WRITE(1,180) (NTRAVEL(I),I=1,NNODES)
170 CONTINUE

    K = 0
    DO 255 I=1,NLINKS
            LN = LTRAVEL(I)
    DO 255 J=1,nb(LN)
    k = k+1
    idb(K) = ibr(LN,j)
255  continue
    nbatt = k

    DO 291 I=1,NLINKS-1
      LN1=LTRAVEL(I)
      LN2=LTRAVEL(I+1)
293     READ(11,292,END=295) LK1,LK2,NOSTR,(ISTR(I,J),J=1,NOSTR)
292     FORMAT(I4,I4,I2,25(I10))
      IF(LK1.EQ.LN1 .AND. LK2.EQ.LN2) THEN
        NB2=NB2+NOSTR
        DO 294 J=1,NOSTR
          K=K+1
          IDB(K) = ISTR(I,J)
C                                         WRITE(*,*)'LK1   ,LK2',LK1,LK2
C                                         WRITE(*,*) ISTR(I,J)
C                                         READ(*,*)
294       CONTINUE
      ENDIF
      GOTO 293
295     REWIND 11
291     CONTINUE

      NBATT=K



c
c    sort
c
     do 260 i=1,nbatt
     do 270  j=i,nbatt
     if (idb(j).lt. idb(i)) then
C      write(*,*)'j',j
      itemp = idb(i)
      idb(i) = idb(j)
      idb(j) = itemp
```

```
      end if
 270   continue
 260   continue

C     WRITE(1,*) 'NBATT=',NBATT
C     do 75 I=1,nbatt
C     write(1,*) idb(i)
C 75  continue

      k = 1
      kk = 2
 277  if (idb(kk) .gt.idb(k)) then
      k = k + 1
      idb(k) = idb(kk)
      else
      kk = kk+1
      end if
      if (kk.le.nbatt) goto 277
      nb2 = k

c**************************************************************
c       converting integer to character

      open(20,file='junk.txt',status='unknown')

                   do 2050 i=1,nb2

                     if (idb(i) .gt. 99999999 ) then
                               write(20,2000) idb(i)
                                       goto 2050
2000       format(1x,i9)
        endif

                     if ( idb(i) .lt. 99999999 .and. idb(i) .gt. 9999999) then
                               write(20,2010) idb(i)
                                       goto 2050
2010       format(1x,'0',i8)
        endif

                     if ( idb(i) .lt. 9999999 ) then
                               write(20,2020) idb(i)
                                       goto 2050
2020       format(1x,'00',i7)
        endif

2050   continue

c*****************************************************************
C     WRITE(*,*) '    NUMBER OF POTENTIAL BRIDGES TO BE CROSSED', NB2
C     WRITE(*,*) '    LENGTH OF TRAVEL PATH (IN MILES)',XXL
      WRITE(1,*)
      WRITE(1,*) '    NUMBER OF POTENTIAL BRIDGES TO BE CROSSED', NB2
      WRITE(1,*) '    LENGTH OF TRAVEL PATH (IN MILES)',XXL
      WRITE(1,*) '    BRIDGE IDENTIFICATION NOS THAT ARE CROSSED :'
      row=row+2
      call settextposition(int2(row),int2(16),curpos)
      call outtext(' NUMBER OF POTENTIAL BRIDGES TO BE CROSSED       ')
      write(str6,'(I6)') nb2
      call settextposition(int2(row),int2(60),curpos)
      call outtext(str6)
      row=row+1
      call settextposition(int2(row),int2(16),curpos)
      call outtext(' LENGTH OF TRAVEL PATH IN MILES                  ')
      write(str10,'(F10.4)') XXL
      call settextposition(int2(row),int2(38),curpos)
      call outtext(str10)
      row=row+1
      col=16
      j=0
```

```
     call settextposition(int2(row),int2(16),curpos)
     call outtext('                                    ')
c     WRITE(*,180) (IDB(I),I=1,NB2)

     rewind 20
1800  read(20,1900,end=1995) c1,C2,C3
     read(20,1900,end=1995) c4,C5,C6
     read(20,1900,end=1995) c7,C8,C9
     read(20,1900,end=1995) c10,C11,C12
1900  format(1x,a4,A2,A3)

     WRITE(1,185) c1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12
185   FORMAT(10x,4(a4,'-',a2,'-',a3,'  '))

        call settextposition(int2(row),int2(16),curpos)
        call outtext(c1)
        call settextposition(int2(row),int2(20),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(21),curpos)
        call outtext(c2)
        call settextposition(int2(row),int2(23),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(24),curpos)
        call outtext(c3)
        call settextposition(int2(row),int2(28),curpos)
        call outtext(c4)
        call settextposition(int2(row),int2(32),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(33),curpos)
        call outtext(c5)
        call settextposition(int2(row),int2(35),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(36),curpos)
        call outtext(c6)

        call settextposition(int2(row),int2(40),curpos)
        call outtext(c7)
        call settextposition(int2(row),int2(44),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(45),curpos)
        call outtext(c8)
        call settextposition(int2(row),int2(47),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(48),curpos)
        call outtext(c9)

                                call settextposition(int2(row),int2(52),curpos)
        call outtext(c10)
        call settextposition(int2(row),int2(56),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(57),curpos)
        call outtext(c11)
        call settextposition(int2(row),int2(59),curpos)
        call outtext('-')
        call settextposition(int2(row),int2(60),curpos)
        call outtext(C12)

c         call settextposition(int2(row),int2(28),curpos)
c         call outtext(cidb2)
c         call settextposition(int2(row),int2(40),curpos)
c         call outtext(cidb3)
c         call settextposition(int2(row),int2(52),curpos)
c         call outtext(cidb4)

         row = row +1
         call settextposition(int2(row),int2(16),curpos)
     call outtext('                                    ')
      if(row.eq.24) then
         row = 4
```

```
          call settextposition(int2(25),int2(29),curpos)
          call outtext(' Press ENTER to continue ')
          pause ' '
          dummy4 = setbkcolor(int2(0))
          call clearscreen($gclearscreen)
          dummy4 = setbkcolor(int2(1))
          call settextposition(int2(row),int2(16),curpos)
      call outtext('                                        ')
        endif
      goto 1800


1995  call settextposition(int2(30),int2(29),curpos)
      call outtext(' Press ENTER to continue ')

      pause ' '


c      PAUSE '                              HIT RETURN TO CONTINUE'
      continue


C
C      OPENING BRINSAP DATA FILE
C
      OPEN(10,FILE='C:\BRINSAP\D12SORT.TXT',STATUS='OLD')
C     OPEN(10,FILE='DIST12.TXT',STATUS='OLD')
      K = 1
                    KK = 1

C//////////////////////////////////////////
      dummy4 = setbkcolor(int4(0))
      call clearscreen($gclearscreen)
      dummy4 = setbkcolor(int4(4))
      call settextposition(int2(5),int2(20),curpos)
      call outtext('                                        ')
      call settextposition(int2(6),int2(20),curpos)
      call outtext(' Bridge Information Report will be generated ')
      call settextposition(int2(7),int2(20),curpos)
      call outtext(' and stored in the file                  ')
      call settextposition(int2(7),int2(46),curpos)
      call outtext(nout)
      call settextposition(int2(8),int2(20),curpos)
      call outtext('                                        ')
      call settextposition(int2(21),int2(29),curpos)
      call outtext(' Press ENTER to continue ')
      pause ' '
      dummy4 = setbkcolor(int4(0))
      call clearscreen($gclearscreen)
      dummy4 = setbkcolor(int4(1))
      call settextposition(int2(12),int2(23),curpos)
      call outtext('     Writing into the file ')
      call settextposition(int2(12),int2(52),curpos)
      call outtext(nout)

C//////////////////////////////////////////
      REWIND 20
390   READ(20,2100,END=5000) CTSEC,CTSECSTR
2100  FORMAT(1X,A6,A3)

 400  CONTINUE
      READ(10,FMT=340,END=4900) A
 340  FORMAT (A514)
      IF (A(7:12) .EQ. CTSEC  .AND. A(18:20) .EQ. CTSECSTR ) THEN
                    GOTO 500
                    ELSE
      GOTO 400
      ENDIF
C
```

```
C  DECODE AND PRINT
C

500   COUNTY= A(4:6)
                     CONT = A(7:10)
                     SEC =  A(11:12)
                     MP  =  A(13:17)
C                    XXL = ICHAR(MP)/1000.
     STNUM = A(18:20)
                     RDES = A(28:28)
     RS = A(30:31)
                     RN = A(32:35)
                     RDIR = A(36:36)
                     FCROS=  A(53:76)
                     CRBR  = A(77:77)
                     FOVER=  A(91:108)
                     LC = A(109:133)
                     VCLEAR = A(134:137)
                     FCLASS = A(171:172)
                     YB = A(173:176)
                     LANON  = A(177:178)
                     LANUN  = A(179:180)
                     DLOAD  = A(189:189)
                     APW  = A(190:192)
                     OPST = A(210:210)
                     LTYPE  = A(211:211)
                     LOAD  = A(212:214)
                     TYSER  = A(215:216)
                     MSTYP  = A(217:220)
                     MJTYP  = A(221:224)
                     MNTYP  = A(225:228)
                     NMSPAN  = A(241:243)
                     NMJSPAN  = A(244:246)
                     NMNSPAN  = A(247:249)
                     TOTSPAN  = A(250:253)
                     HCLEAR  = A(254:257)
                     MXSPAN  = A(258:261)
                     STRLEN  = A(262:267)
                     RWIDE  = A(274:277)
                     DCKWID=A(278:281)
                     DCKCOND=A(303:303)
                     SPRCOND= A(304:304)
                     SBCOND  = A(305:305)
                     OPRAT  = A(308:310)
                     IRATING  = A(312:314)
                     STREVA  = A(315:315)
                     SLCAP  = A(318:318)
                     DEFHD  = A(408:408)
     if(ipcode.eq.1 .or. ipcode.eq.2) then
c                    write(*,600) COUNTY,STNUM,RS,RN,RDIR
c                    WRITE(*,601) FCROS,FOVER,LC
c                    WRITE(*,602) LANON,DLOAD,OPST,LTYPE,LOAD,MSTYP
c                    WRITE(*,603) NMSPAN,TOTSPAN,MXSPAN,STRLEN,RWIDE
c                    WRITE(*,604) OPRAT,IRATING,DEFHD


                     write(1,600) COUNTY,STNUM,RS,RN,RDIR
                     WRITE(1,601) FCROS,FOVER,LC
                     WRITE(1,602) LANON,DLOAD,OPST,LTYPE,LOAD,MSTYP
                     WRITE(1,603) NMSPAN,TOTSPAN,MXSPAN,STRLEN,RWIDE
                     WRITE(1,604) OPRAT,IRATING,DEFHD

 600  FORMAT (1X,60('*'),/,1X,'CNTY ',A3,' STR NO. ',A3,'  R. S. ',A2,
    & '  R No. ',A4,'  R. DIR  ',A1)
 601  FORMAT (1X,' FEATURE CROSSED ---- ',A24,/,
    &        1X,' FAC. CARRIED OVER -- ',A18,/,
    &        1X,' LOCATION OF BR. ---- ',A25)
 602  FORMAT (1X,' No. LANES ',A2,'  DESIGN LOAD  ', A1,'  OP. STAT. '
    &   ,A1,/,1X,'  LOAD TYPE ',A1,'  LOAD ', A3, '  MAIN SPAN TYPE ',
```

```
  &  A4)
603 FORMAT ( 1X, ' No. MAIN SPANS ',A3,' TOTAL No. SPANS ',A4,
  & '  MAX. SPAN LGTH ',A4,/,1X,'  STRUC. LENGTH ',A6,
  & ' ROAD WIDTH  ', A4)
604 FORMAT (1X, ' RATINGS **** OPERATION ', A3,'   INVENTORY ',A3,
  & /,1X, '  DEFENSE HWY DESIGN ', A1)
   GOTO 390
                 ENDIF

4900 REWIND 10
                 GOTO 390
5000 CONTINUE

   dummy4 = setbkcolor(int4(0))
   call clearscreen($gclearscreen)

5001 end
```