Technical Report Documentation Page

1. Report No. TX-95/1909-1F	2. Government Accession No.	3. Recipient's Catalog No.

4. Title and Subtitle
TEXAS REFERENCE MARKER AUTOMATED RI-1 DIAGRAMS

5. Report Date
January 1993:
Revised April 1993

6. Performing Organization Code

7. Author(s)
Saïd Majdi, Anita Kalamdani, and Hassan Charara

8. Performing Organization Report No.
Research Report 1909-1F

9. Performing Organization Name and Address
Texas Transportation Institute
The Texas A&M University System
College Station, Texas 77843-3135

10. Work Unit No. (TRAIS)

11. Contract or Grant No.
Study No. 7-1909

12. Sponsoring Agency Name and Address
Texas Department of Transportation
Research and Technology Transfer Office
P. O. Box 5080
Austin, Texas 78763-5080

13. Type of Report and Period Covered

14. Sponsoring Agency Code

15. Supplementary Notes
Research performed in cooperation with the Texas Department of Transportation
Research Study Title: Texas Reference Marker -- Automated RI-1 Diagrams

16. Abstract
The Automated Road Inventory (ARI) is a software application used to automate the generation of Roadway Inventory (RI-1) diagrams. The RI-1 diagrams, which are hand drawn visual roadway records created by the Division of Transportation Planning of the Texas Department of Transportation (TxDOT), depict all highways maintained by the Department. ARI provides full automation to generate a hard copy of the diagrams from an ASCII input data file that contains a roadway features description. The ARI process was developed by the Texas Transportation Institute (TTI) as a system of C, UCM (User Command Macro), and MDL (MicroStation Development Language) programs that process roadway feature text data into graphical drawings. The process was developed within the MicroStation 4.0 environment and is designed with built-in flexibility in order to make customization easy. TxDOT will use a 486/33 MHZ ISA microcomputer with 8 MB of RAM and 200 MB hard drive to run ARI. ARI is menu-driven and provides several clear and easy menus which allow the user to select from different available options, create RI-1 diagrams, preview them, and send them to either a local or network plotter. The customization ability of ARI opens the door to a multitude of other possible applications that require graphical representation of textual database.

17. Key Words
RI-1, Roadway Inventory, Graphics, Automated Diagrams

18. Distribution Statement
No restrictions. This document is available to the public through NTIS:
National Technical Information Service
5285 Port Royal Road
Springfield, Virginia 22161

19. Security Classif.(of this report) Unclassified	20. Security Classif.(of this page) Unclassified	21. No. of Pages 92	22. Price

TEXAS REFERENCE MARKER AUTOMATED RI-1 DIAGRAMS

by

Saïd Majdi
Assistant Research Scientist
Texas Transportation Institute

Anita Kalamdani
Assistant Research Scientist
Texas Transportation Institute

and

Hassan Charara
Assistant Research Scientist
Texas Transportation Institute

Research Report 1909-1F
Research Study Number 7-1909
Research Study Title: Texas Reference Marker -- Automated RI-1 Diagrams

Sponsored by the
Texas Department of Transportation

January 1993
Revised: April 1993

TEXAS TRANSPORTATION INSTITUTE
The Texas A&M University System
College Station, Texas 77843-3135

IMPLEMENTATION STATEMENT

This study recommends the use of the Automated Roadway Inventory (ARI) software for the generation of RI-1 diagrams. ARI provides full automation to generate a hard copy of diagrams from an ASCII input data file that contains a roadway features description. The ARI process was developed within the MicroStation 4.0 environment and will be implemented on a 486/33 MHZ ISA microcomputer with 8 MB of RAM and 200 MB hard drive. ARI uses the MicroStation 4.0 graphic environment in a seamless fashion to provide a friendly graphic user interface.

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 Texas Department of Transportation. This report does not constitute a standard, specification, or regulation, nor is it meant for construction, bidding, or permit purposes.

ACKNOWLEDGMENT

This research was sponsored by the Texas Department of Transportation. The authors wish to thank Messrs. Larry Kitten, Joel Young, Vennie Davis, and Jim Copeland of TxDOT for their guidance during this study. A special thanks goes to the staff of the Texas Reference Marker (TRM) team. In particular, Mr. Michael Preston was instrumental in supplying technical data; his efforts are sincerely appreciated.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

Figure **Page**

SUMMARY

The Automated Road Inventory (ARI) is a software application that runs within the MicroStation PC 4.0 environment. The purpose for the development of ARI is to automate the generation of the RI-1 Diagram which is a hand drawn visual roadway record created by the Division of Transportation Planning of TxDOT to depict all highways maintained by the Department. The Texas Reference Marker Project (TRM) initiated the effort to establish a uniform identification key for all roadway related files, revise the Roadway Record Database, and automate the generation of the RI-1 diagram. The Texas Transportation Institute (TTI) developed ARI as a system of C, UCM (User Command Macro), and MDL (MicroStation Development Language) programs that process roadway feature text data into graphical drawings. The ARI process is menu-driven. Several clear and easy menus are provided to the user to select from the different available options, create RI-1 diagrams, preview them, and send them to a local or network plotter. ARI is designed with great built-in flexibility in order to make customization easy. TxDOT will use a 486/33 MHz ISA microcomputer with 8 MB of RAM and 200 MB hard drive to run ARI. A full description of the ARI process, as well as all the menus and commands the user needs to run ARI and to generate RI-1 diagrams, is included.

1.0 INTRODUCTION AND DESCRIPTION OF THE ARI PROCESS

1.1 INTRODUCTION

The Automated Road Inventory (ARI) software application was developed to automate the generation of the RI-1 diagrams, a visual roadway record created by the Division of Transportation Planning of TxDOT. ARI runs within the MicroStation PC 4.0 environment. At the beginning of the development of the ARI software, all the different modules were hard-coded in C, including the forms and the graphic symbols. After the project began and after a few specifications modifications were made by TxDOT, it was apparent that a major change in the software design approach had to be made. A description format was devised to describe all input as well as output elements. Symbol Description Format (SDF) files were then written to describe the different forms and graphic symbols. The SDF files are in ASCII; therefore, they are very easily generated and edited by non-programmers in case a modification is required. A C 'kernel' compiles the different SDF files needed for the generation of RI-1 diagrams. A full description of SDF will be given later in this document.

1.2 THE INPUT DATA FILE

The ARI process starts with a data file that contains the description of the roadway features in the highway section to be processed. This data file must reside in the proper directory on the user's computer. This data file can be obtained in two ways: (1) by loading it from a diskette, or (2) by downloading from the department's mainframe through a communication link. It is TxDOT's intention to have users communicate with the mainframe through Ethernet and make a request for the data needed to generate ARI diagrams. The communications and query processing have been handled by the Texas Reference Marker) TRM group of the TxDOT Automation Division. TTI and the TRM team have developed a format for the input data file that can describe all highway features. Figure 1 shows the different fields of the six types of records used to describe the highway features that will become graphic drawing elements after processing.

1.3 ARI DIAGRAM GENERATION

When the ARI process is initiated, MicroStation PC is loaded with an ARI menu that has all the commands necessary to generate, preview, and plot RI-1 diagrams. First, an initialization is performed and certain parameters are set to their default values. Namely, the following parameters are initialized:

- *Number of miles per page (mpp)* which determines the length of roadbed represented in each diagram and, therefore, determines the number of diagrams that will be generated of a highway section of a certain length

- *Mile increment (mincr)* which represents the number of tick marks along the roadbed

(0 = HEADER SHEET/SHEET HEADERS/TRAILERS)

SHEET NUMBER	BLANK	PLOT SCALE	BLANK	ROADBED ID	BLANK	SYMBOL NUMBER	BLANK	RECORD TYPE	BLANK	NOTATION FIELD 1
2C		8C		1C		5C		1C		35C

(1 = TEXT ONLY VALUES)

SHEET NUMBER	BLANK	PLOT SCALE	BLANK	ROADBED ID	BLANK	SYMBOL NUMBER	BLANK	RECORD TYPE	BLANK	REFERENCE MARKER	BLANK	DISPLACEMENT	BLANK	DISTANCE FROM ORIGIN	BLANK	NOTATION FIELD 1
2C		8C		1C		5C		1C		5C		7C		8C		35C

(2 = SYMBOLS ONLY NO TEXT)

SHEET NUMBER	BLANK	PLOT SCALE	BLANK	ROADBED ID	BLANK	SYMBOL NUMBER	BLANK	RECORD TYPE	BLANK	REFERENCE MARKER	BLANK	DISPLACEMENT	BLANK	DISTANCE FROM ORIGIN
2C		8C		1C		5C		1C		5C		7C		8C

(3 = SYMBOLS WITH TEXT/SKEW)

SHEET NUMBER	BLANK	PLOT SCALE	BLANK	ROADBED ID	BLANK	SYMBOL NUMBER	BLANK	RECORD TYPE	BLANK	REFERENCE MARKER	BLANK	DISPLACEMENT	BLANK	DISTANCE FROM ORIGIN	BLANK	SKEW	BLANK	NOTATION FIELD 1	BLANK	NOTATION FIELD 2
2C		8 C		1C		5C		1C		5C		7C		8C		4C		35C		35C

(4 = SYMBOLS WITH TEXT NO SKEW)

SHEET NUMBER	BLANK	PLOT SCALE	BLANK	ROADBED ID	BLANK	SYMBOL NUMBER	BLANK	RECORD TYPE	BLANK	REFERENCE MARKER	BLANK	DISPLACEMENT	BLANK	DISTANCE FROM ORIGIN	BLANK	NOTATION FIELD 1	BLANK	NOTATION FIELD 2	BLANK	NOTATION FIELD 3	BLANK	NOTATION FIELD 4
2C		8C		1C		5C		1C		5C		7C		8C		35C		35C		35C		35C

(5 = SUMMARY RECORDS)

SHEET NUMBER	BLANK	PLOT SCALE	BLANK	ROADBED ID	BLANK	SYMBOL NUMBER	BLANK	RECORD TYPE	BLANK	REFERENCE MARKER	BLANK	DISPLACEMENT	BLANK	DISTANCE FROM ORIGIN	BLANK	NOTATION FIELD 1	BLANK	VALUE FIELD 1	BLANK	NOTATION FIELD 2	BLANK	VALUE FIELD 2	BLANK	NOTATION FIELD 3	BLANK	VALUE FIELD 3
2C		8C		1C		5C		1C		5C		7C		8C		35C		35C		35C		35C		35C		35C

Note: C := Character

Figure 1. Automated RI-1 Input Data Format

- *DFO/DISP* parameter which determines the method of locating features along the roadbed: Distance From Origin or DISPlacement from previous reference marker

- *Plotter* which determines the device that will be used to generate the diagrams

The user has, of course, the option of changing the default settings before proceeding.

After initialization, the user is prompted to load an input data file. The records contained in this data file are then processed to create the corresponding graphic elements. When the creation of the graphic elements is complete, the RI-1 diagrams are displayed on the computer screen for the user to preview. By selection of the command, Plot, the user is able to send the diagrams to a plotter for the purpose of generating a hard copy. This terminates the ARI diagram generation. The user can repeat this process as many times as desired by using the ARI menu commands and following the steps of loading an input data file, generating diagrams, viewing them, and sending them to the plotter. Figure 2 shows the flowchart that describes the ARI diagram generation process.

1.4 DATA TRANSLATION - THE ARI PROCESS

Data translation consists of reading each record in the input data file and generating the corresponding graphic element. This process involves two subprocesses. The first is Pass 1 or clutter analysis, and the second is Pass 2 or roadway feature processing and placement (see flowchart in Figure 3).

Pass 1 is necessitated by the fact that the graphic symbols representing the roadway features have a fixed size, whereas the space where they are to be placed, i.e., miles per page (mpp), can vary. This might cause an overlap of symbols in near proximity of each other. Some symbols such as boundary symbols are allowed to overlap since they only cause a no clutter overlap or NCO. The set of symbols that might cause NCO are defined in an ASCII file which is read before the start of Pass 1. Clutter analysis is performed by comparing coordinates of envelopes of two successive feature symbols. By determining the magnitude of the overlap, an optimal miles per page number, mpp*, is calculated. The value in mpp* represents the largest number of miles per page that will prevent clutter from occurring. The smallest value that mpp can have is 0.5 miles. Because of this limit, which is set by the Department's specifications, clutter of graphic symbols can still occur in diagrams generated at mpp=0.5. This situation is, however, considered to be very unlikely.

In Pass 1, when clutter is detected, the user is prompted to choose between manual and automatic processing of clutter. If manual processing is chosen, mpp is forced to have the user's setting. If automatic processing is chosen, the calculated mpp* is used. A flowchart describing Pass 1 is shown in Figure 4.

Pass 2, or feature processing and placement, consists of the creation of the actual RI-1 diagrams. The process is started with the initialization of the global dimensions of the diagrams; then, tables that describe the relationships between the different elements of the diagrams are loaded. At

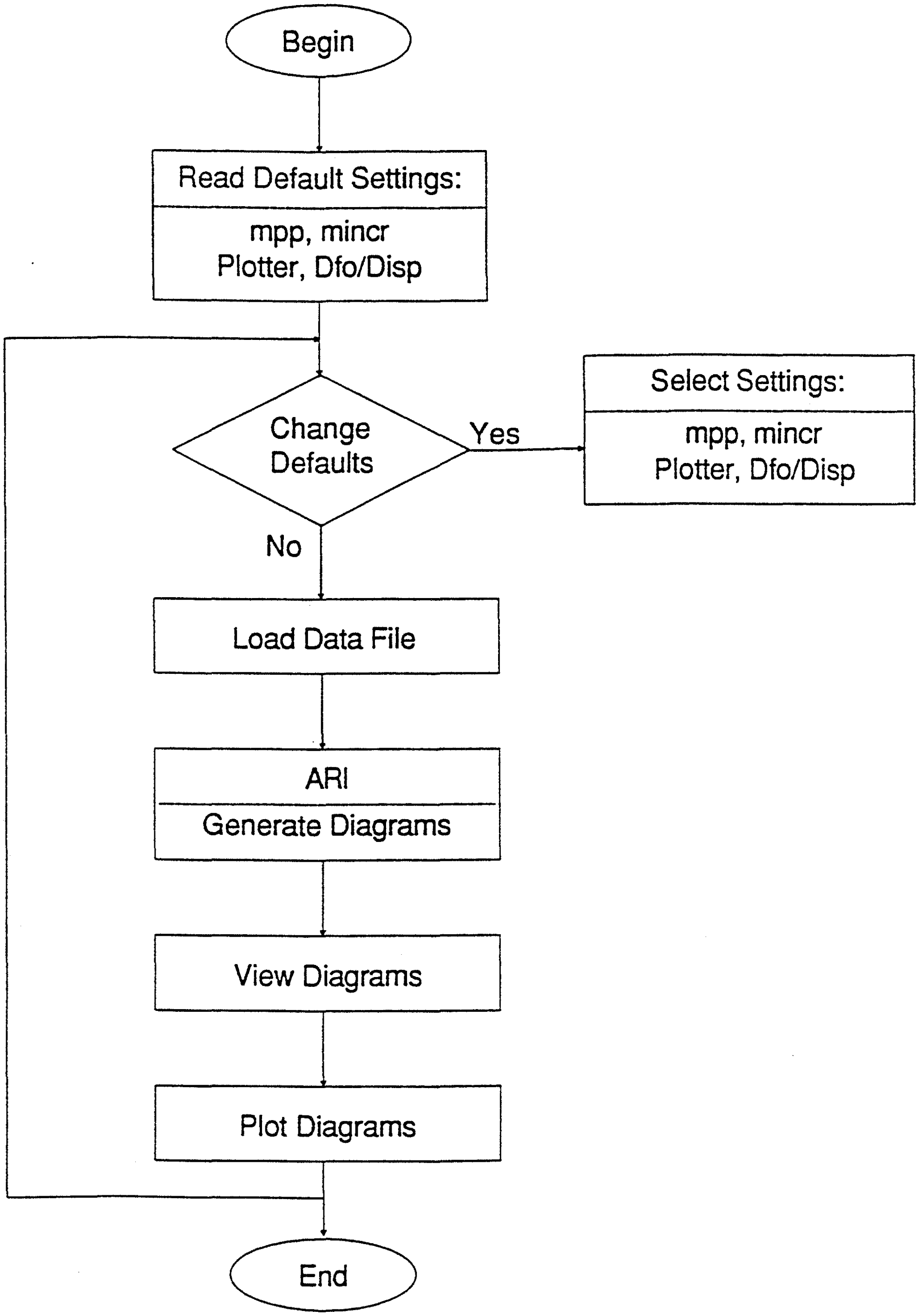


Figure 2. Flowchart of the ARI Diagram Generation Process

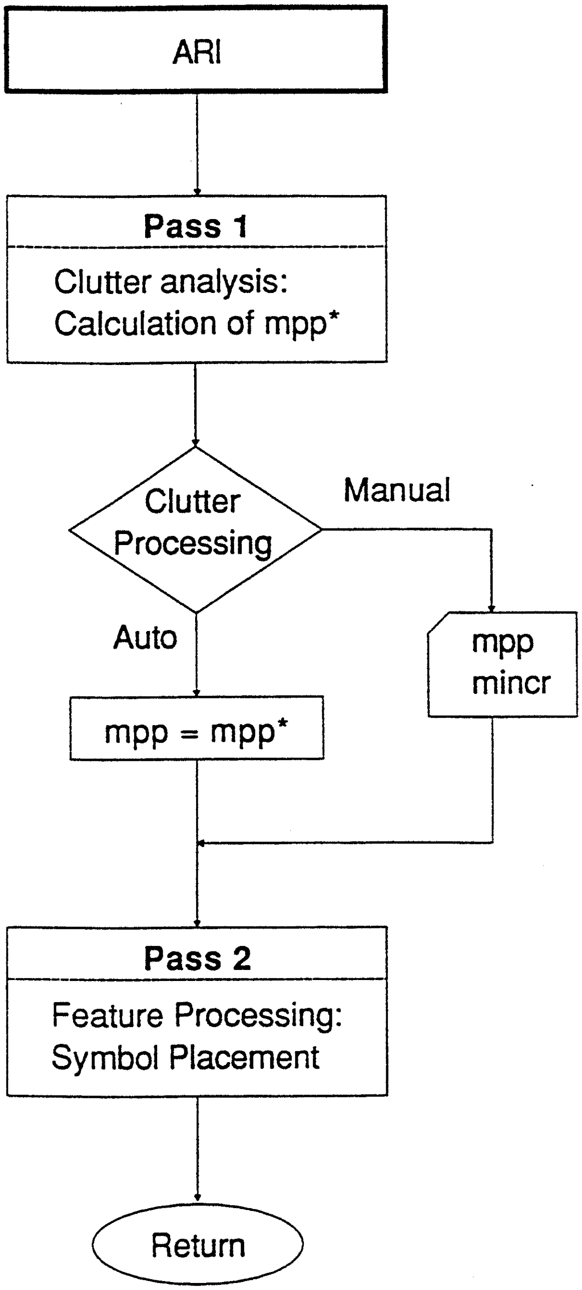


Figure 3. Flowchart of the ARI Data Translation Process

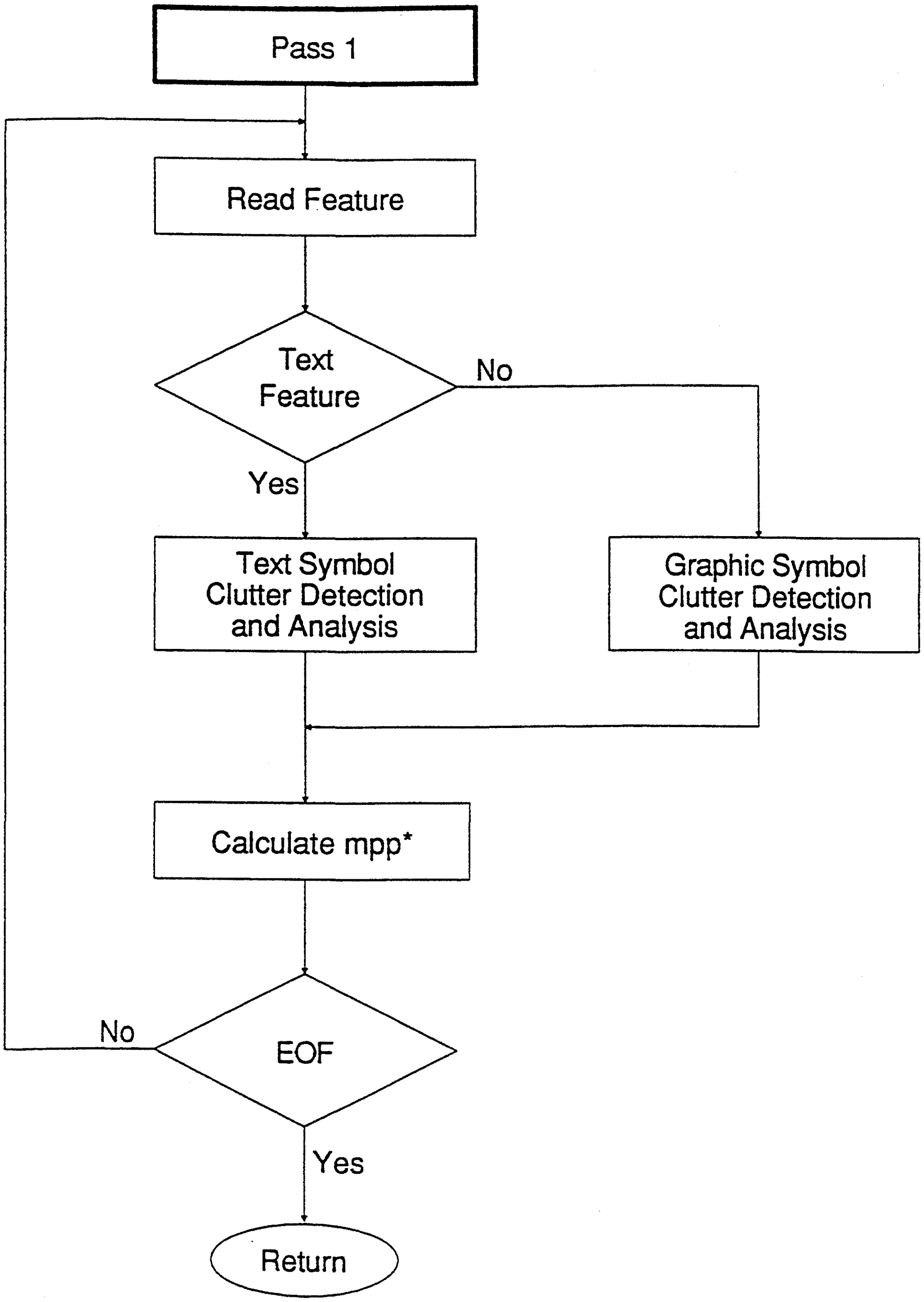

Pass 1
Read Feature
Text Feature
No
Yes
Text Symbol Clutter Detection and Analysis
Graphic Symbol Clutter Detection and Analysis
Calculate mpp*
EOF
No
Yes
Return

Figure 4. Pass 1 - Calculation of the Optimal mpp

this point, ARI is ready to translate records from the input data file into a graphic description. The format used for this graphic description is Drawing Interchange Format (DXF). DXF files are ASCII files that contain text describing graphic elements (such as line, arc, circle). Most CAD software packages have the ability to translate from and into DXF. As the name indicates, this standard format allows the exchange of drawing files between the different CAD packages even if they don't use the same format, as long as they have a DXF translator. The flowchart that describes Pass 2 is shown in Figure 5.

After the creation of the DXF file, the MicroStation resident DXF translator is used to convert the DXF file into a MicroStation DGN file and automatically display it on the screen. The user is then able to preview the diagrams using MicroStation commands such as view, zoom in/out, move up/down, and move left/right.

1.4.1 Forms

There are basically four B-size (17" by 11") forms needed for the generation of ARI diagrams:

- **The Plot Report Form**. This is the first page to be plotted. This form contains information from the input data file such as data date, diagram limits, diagram format and data limiters, as well as information generated by the ARI process, such as output exclusions and sheet summary. Figure 6 shows the format of the plot report form.

- **The K, N, L/R, M/S, and P/T Roadbeds Form**. This form and the next contain visual representation of the roadway features in the form of graphic and text symbols. The format of this form is shown in Figure 7.

- **The X/A and Y/B Roadbeds Form**. Both this and the previous form contain information extracted from the input data file such as roadway composition, limits, design data, maintenance data, administrative data, and identification, in addition to some footer information such as highway name, starting and ending reference markers, data date, and sheet number. The format of this form is shown in Figure 8.

- **The Interchanges Form**. This form is a frame with the same footer information contained in the two previous forms. It will enclose the drawing of the interchange defined in the corresponding record of the input data file. Figure 9 shows the format of the interchange form.

1.4.2 Graphic Symbols

In order to make the processing easy, modification and addition of new graphic symbols, and a description format was devised for graphic symbols as well as other drawing elements necessary for the generation of RI-1 diagrams. Symbol Description Format (SDF) files were then written to

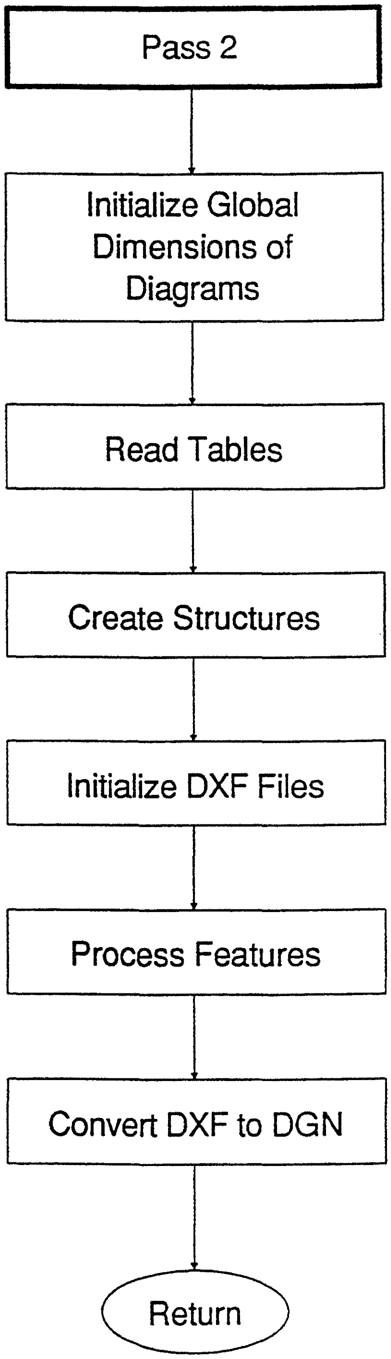
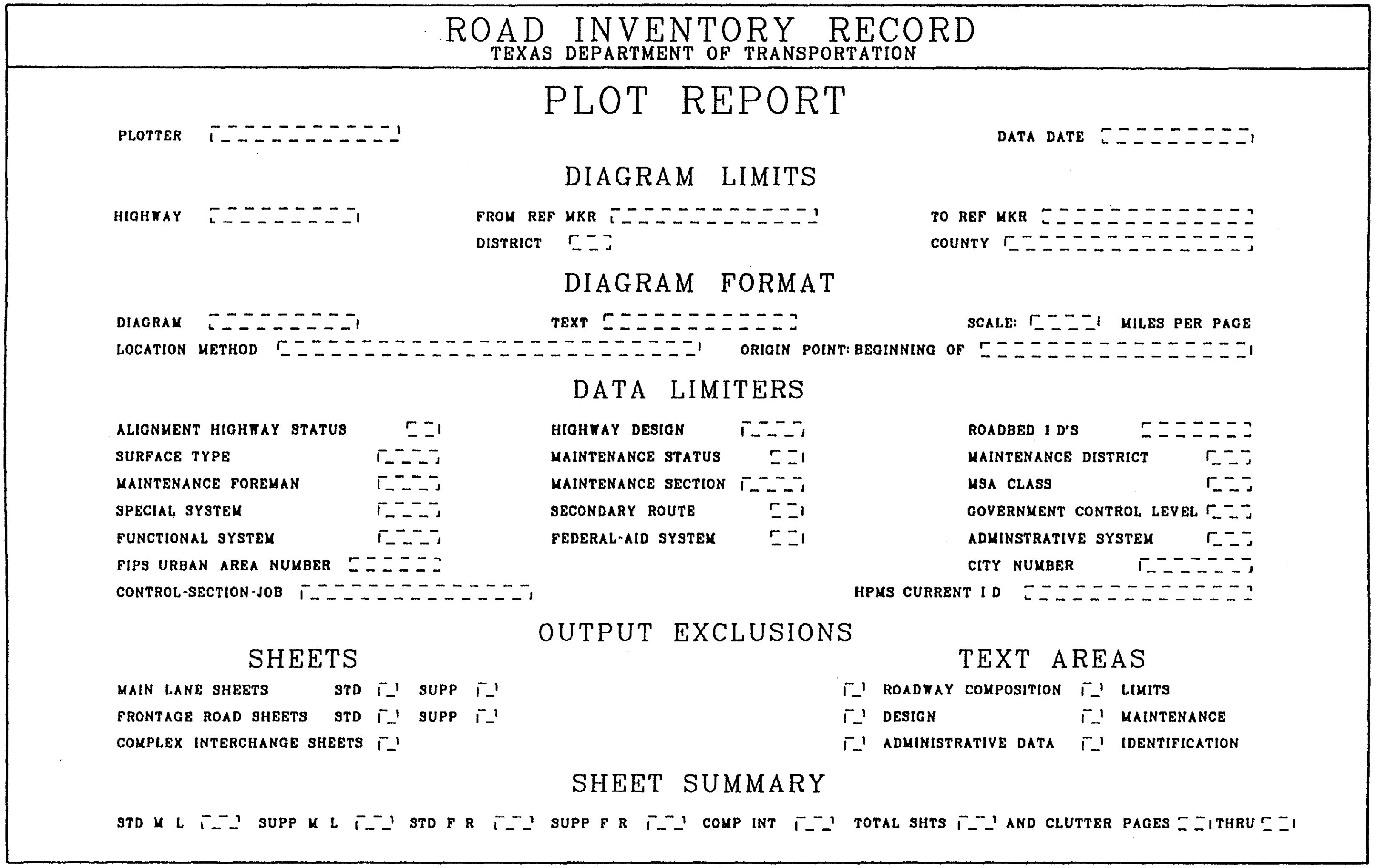


Figure 5. Pass 2 - Generation of the DGN File



ROAD INVENTORY RECORD
TEXAS DEPARTMENT OF TRANSPORTATION

PLOT REPORT

PLOTTER ____________ DATA DATE __________

DIAGRAM LIMITS

HIGHWAY __________ FROM REF MKR ______________ TO REF MKR ______________
DISTRICT ___ COUNTY ________________

DIAGRAM FORMAT

DIAGRAM __________ TEXT _____________ SCALE: ____ MILES PER PAGE
LOCATION METHOD ____________________________ ORIGIN POINT: BEGINNING OF __________________

DATA LIMITERS

ALIGNMENT HIGHWAY STATUS __ HIGHWAY DESIGN ____ ROADBED I D'S _______
SURFACE TYPE ____ MAINTENANCE STATUS __ MAINTENANCE DISTRICT ___
MAINTENANCE FOREMAN ____ MAINTENANCE SECTION ____ MSA CLASS ___
SPECIAL SYSTEM ____ SECONDARY ROUTE __ GOVERNMENT CONTROL LEVEL ___
FUNCTIONAL SYSTEM ____ FEDERAL-AID SYSTEM __ ADMINSTRATIVE SYSTEM ___
FIPS URBAN AREA NUMBER ______ CITY NUMBER _______
CONTROL-SECTION-JOB ________________ HPMS CURRENT I D _______________

OUTPUT EXCLUSIONS

SHEETS

MAIN LANE SHEETS STD □ SUPP □
FRONTAGE ROAD SHEETS STD □ SUPP □
COMPLEX INTERCHANGE SHEETS □

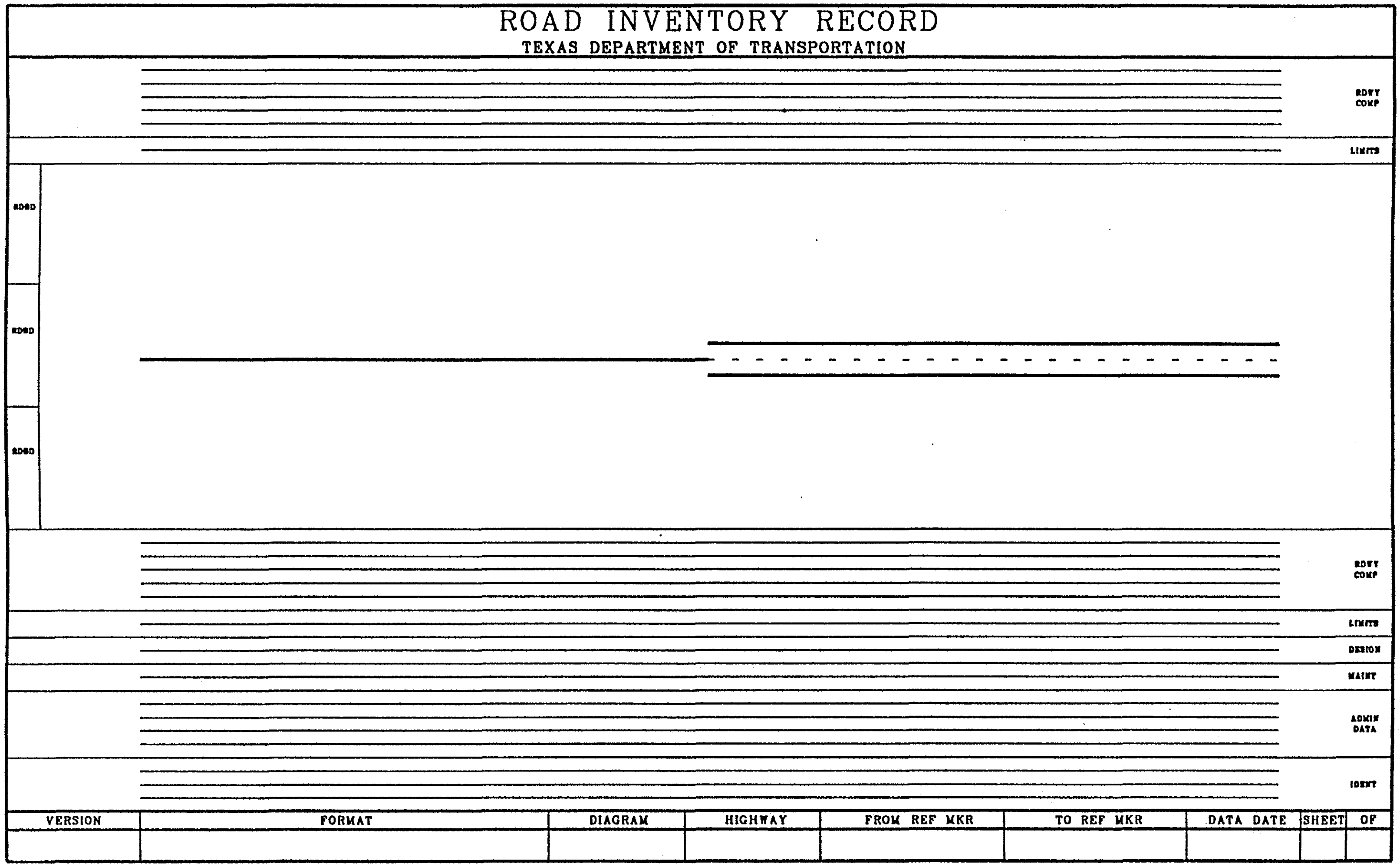
TEXT AREAS

□ ROADWAY COMPOSITION □ LIMITS
□ DESIGN □ MAINTENANCE
□ ADMINISTRATIVE DATA □ IDENTIFICATION

SHEET SUMMARY

STD M L ___ SUPP M L ___ STD F R ___ SUPP F R ___ COMP INT ___ TOTAL SHTS ___ AND CLUTTER PAGES __ THRU __

Figure 6. The Plot Report Form



ROAD INVENTORY RECORD

TEXAS DEPARTMENT OF TRANSPORTATION

RDWY COMP

LIMITS

RDBD

RDBD

RDBD

RDWY COMP

LIMITS

DESIGN

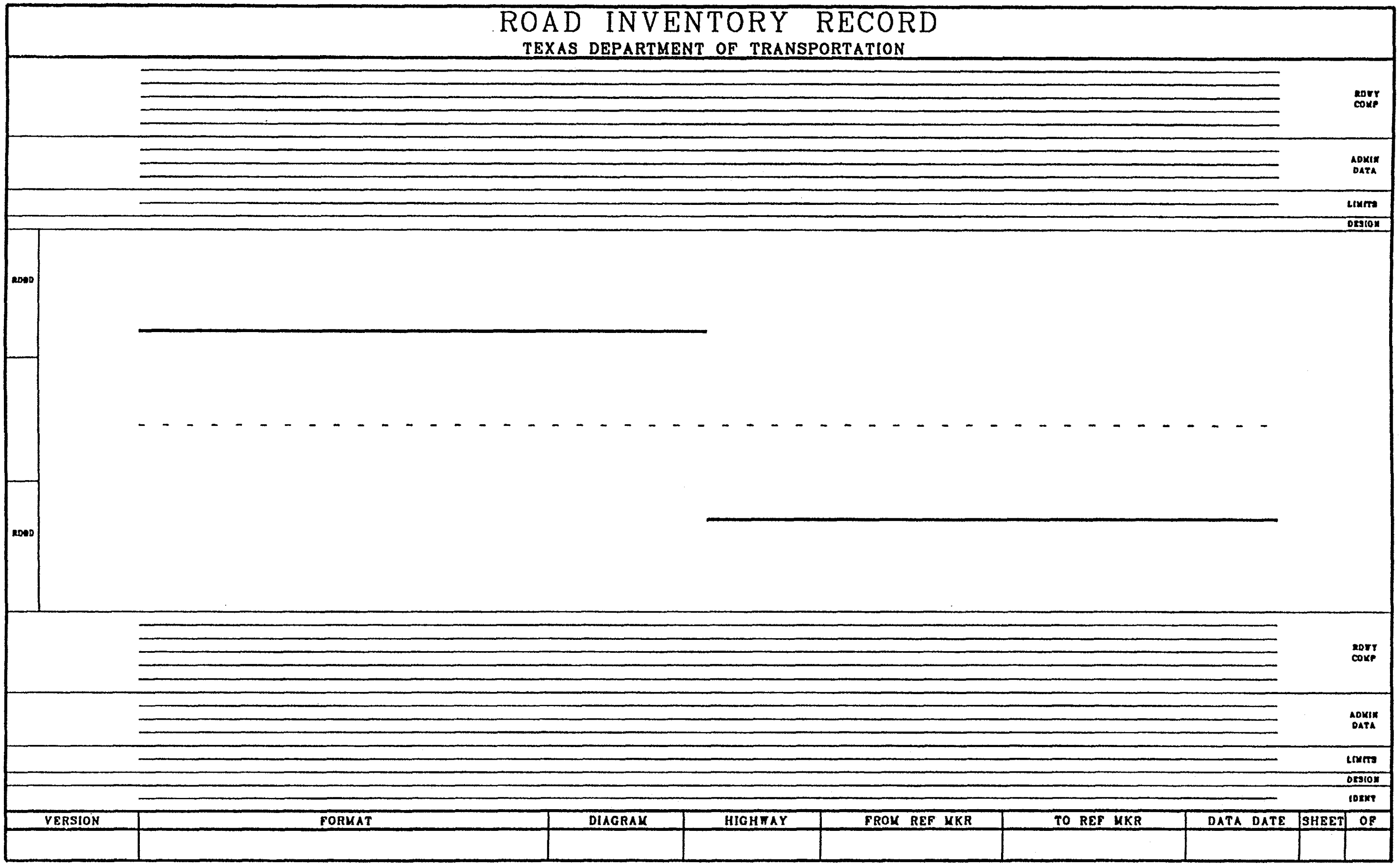
MAINT

ADMIN DATA

IDENT

VERSION	FORMAT	DIAGRAM	HIGHWAY	FROM REF MKR	TO REF MKR	DATA DATE	SHEET	OF

Figure 7. Form for K, N, L/R, M/S, and P/T Roadbeds



ROAD INVENTORY RECORD

TEXAS DEPARTMENT OF TRANSPORTATION

RDWY COMP
ADMIN DATA
LIMITS
DESIGN
RDBD
RDBD
RDWY COMP
ADMIN DATA
LIMITS
DESIGN
IDENT

VERSION	FORMAT	DIAGRAM	HIGHWAY	FROM REF MKR	TO REF MKR	DATA DATE	SHEET	OF

Figure 8. Form for X/A and Y/B Roadbeds

ROAD INVENTORY RECORD

TEXAS DEPARTMENT OF TRANSPORTATION

VERSION	FORMAT	DIAGRAM	HIGHWAY	FROM REF MKR	TO REF MKR	DATA DATE	SHEET	OF
	INTERCHANGE							

Figure 9. Form for Interchange Drawing

describe all the graphic symbols. The SDF files are in ASCII. Therefore, they may be very easily generated and edited by non-programmers if modifications are required or new symbols must be added. A C 'kernel' compiles the different SDF files needed for the generation of RI-1 diagrams. The definition of the SDF format is given in Appendix A.

Some rules to be observed when creating and naming SDF files include:

- When defining the text elements of an SDF file, all the 'V' type text should be before the 'N' type text.

- Whenever the text size is not specified, the default size of 0.0625" is used.

- All the SDF entities should be defined considering the origin (0,0) to be located on the roadbed.

- Symbols crossing the roadbed should be named <symbolno><rdbd>C.sdf; for example, BS111KC.SDF is the filename for symbol number BS111, on roadbed K, and crossing the K line.

- Symbols in the first or second quadrant should be named <symbolno><rdbd>A.sdf; for example, BS111KA.SDF is the filename for symbol number BS111, on roadbed K, and located above the K line.

- Symbols in the third or fourth quadrant should be named <symbolno><rdbd>B.sdf; for example, BS111KB.SDF is the filename for symbol number BS111, on roadbed K, and located below the K line.

The variable <rdbd> represents the roadbed name which can be equal to K for a single roadbed roadway or L, M, P, X, or Y and R, S, T, A, or B for a multiple roadbed roadway.

In the ARI data file, symbols are defined with a skew. The skew is specified by a 4-digit numeric code:

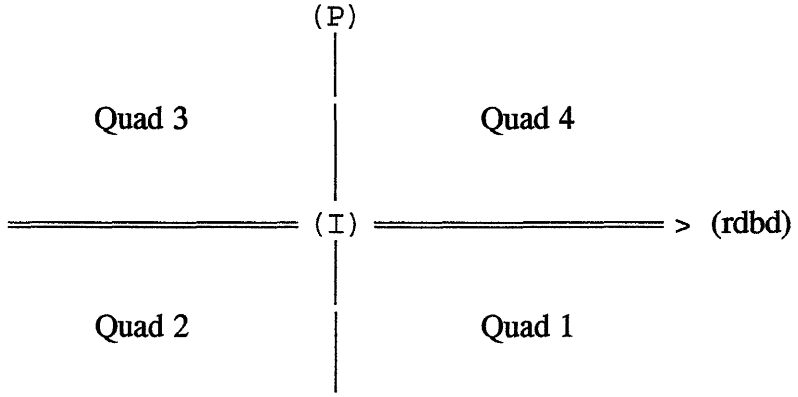
QIAA

where:

Q : Quadrant number (Q=1, 2, 3, or 4)

I : I=1 if crossing; I=0 if not crossing the roadbed

AA : Angle of intersection with roadbed (AA is an integer between 01 and 90)



If skew is not defined (record type 2), then the symbol's SDF filename is the same as that for the crossing symbol.

In processing a data record which describes a graphic feature, the SDF filename is generated based on the information contained in that record. The SDF file is then processed line by line, and the corresponding drawing element is placed in a DXF file.

1.4.3 Text Symbols

Text symbols are divided into two categories: simple text features and composite text features. Simple text features contain a notation and a value. Composite text features are combinations of simple text features. After reading a text feature data record from the data file, the following steps are executed in order to have the corresponding text placed in the RI-1 diagram:

1. Determine if the text symbol is simple or composite by reading the feature code.

2. If the text symbol is simple, then use the NV (Notation/Value) table to determine if notation should come before value (n+v) or vice-versa (v+n), and compose appropriately.

3. Determine from the FA (Feature/Area) tables the location(s) in the different forms where text is to be placed and create the corresponding DXF code.

4. If the text symbol is composite, then decompose the symbol into simple text features and process each simple text feature individually as described above. After processing the simple text features, concatenate them to form the text corresponding to the composite text feature, then complete step 3.

1.4.4 Interchanges

Interchanges are described in the data file by a record that has the interchange number. This is a unique nine-digit code; for example, 146600283 represents interchange 00283 of type 66 in district 14. Because DOS does not allow using more than 8 characters for filenames, the files containing interchange drawings are named using the last five digits only without the four digits that represent the district and the type. However, the whole number (all 9 digits) is plotted with the interchange in the corresponding sheet of the RI-1 diagrams.

1.4.4.1 Creating Interchange Drawing Files

There are two categories of interchanges: standard and complex. Complex interchanges are interchanges with a unique geometry that depends on their location in the highway network. These interchanges are usually hand drawn and, therefore, need to be put in a drawing file through the process of digitization. Standard interchanges are simpler and of the more common type such as T, Y, and diamond. The drawing files for these interchanges are created using MicroStation.

1.4.4.2 Standard Interchanges

Twenty-seven standard interchanges of various types were drawn using MicroStation. These interchanges represent the most commonly found interchanges in the state. The following is a list of the types of standard interchanges for which a drawing file was created. Some standard interchange types have more than one configuration. The number of configurations for a specific type is given under Count in the table below.

Type	Name	Count
1	Y	4
2	T	4
3	X	3
4	Partial Cloverleaf	6
5	Cloverleaf	2
6	Diamond	4
7	Semi-Directional	2
8	Directional	2

Total: 27

1.4.4.3 Complex Interchanges

Complex interchanges are hand drawn by district staff members and then submitted to D-10. In order to include complex interchange drawings in the automated RI-1 diagrams, the hand drawn interchanges were digitized into MicroStation design files using a digitizing tablet. The following steps describe how a complex interchange drawing is prepared for the automated RI-1 process.

1. Draw or digitize a complex interchange into <filename>.dgn.

2. Attach a cell library (e.g., int.cel) and create a cell out of the interchange by fencing the drawing and defining the center of the drawing as the origin.

3. Open a new design file.

4. Attach the same cell library (i.e., int.cel) and place the cell that was created in step 2.

5. Load MDL utility **dxfout** and convert this cell file into a DXF file.

The following is a step-by-step description of the DXF conversion needed for the creation of interchange files ready to be included in automated RI-1 diagrams.

1. At the Ustn prompt, type **place fence block** and press **Enter** or select **place fence** from the Place Fence menu. Place a fence as big as the interchange drawing so all drawing elements are comprised in the fence.

2. Select **Cell Library** from the File menu. This command is used to create a new cell library or to attach an existing library.

3. Type **define cell origin** and press **Enter**. This will define an origin for the cell. The origin is defined by clicking in the center of the drawing or by typing **xy=<x>,<y>** and pressing **Enter**. <x> and <y> are the desired coordinates of the cell origin.

4. Type **cc=<cell name>** and press **Enter**. This creates a cell in an attached library. If interchange 146600283 is being processed, <cell name> will be 00283.cel.

5. Select **Cells** from the Settings menu. This will show all the cells present in an attached cell library.

6. Open a new design file by selecting **New** from the File menu.

7. Attach the library which contains the just created cell by selecting **Cell library** from the File menu.

8. Type **ac=<cell name>** and press **Enter**. This will bring up the cell and place it in the center of the screen.

9. Convert the cell file into a DXF file by typing **mdl load dxfout** and pressing **Enter**, or by using the MDL Applications menu.

All the created DXF files of the interchanges must reside in the \ri1data\ichange directory.

2.0 USER INTERFACE

2.1 PRIMARY DESIGN OBJECTIVE

The primary design objective of the user interface is to minimize user interaction with ARI. The user interface is designed so that even users unfamiliar with the MicroStation 4.0 environment can run ARI. This objective was achieved by designing a simple, yet comprehensive, menu. Dialog boxes have also been made clear and informative in order to facilitate user selection. The following is a description of the user interface as well as the different commands necessary to run the ARI process from logging on to generating a hard copy of the RI-1 diagrams and logging off.

2.2 LOGGING ON TO ARI

Users may follow one of the two steps listed below to start ARI.

1. From the DOS C:\ or C:\(Any Other Directory) prompt:
   Type **cd\ustation** and press **Enter**. The current directory changes to **C:\ustation**. Type **ari** and press **Enter**. The **ari** command will load MicroStation with the **ARI** application attached to it.

2. From the MicroStation command line:
   Type **mdl load ari** and press **Enter**. The **ARI** application will be loaded and the **ARI** menu attached.

2.3 THE ARI MENU

When selected, in addition to the menus that MicroStation is configured to have initially loaded, the **ARI** menu, depicted in Figure 10, is opened in the upper-right corner of the screen. The **ARI** menu provides the user with the commands needed to run the **ARI** process. The **ARI** menu commands are:

- **Create** brings up the **Create** window which allows selection of the diagram parameters and the data file to process.

- **Run** starts the processing of the data file and the actual generation of RI-1 diagrams.

- **View** brings up a dialog box for the selection of display parameters: DFO for distance from origin, DIS for marker displacement, or DFODIS for the display of both DFO and DIS.

- **Plot** brings up a dialog box for the selection of the type of plotter to be used. It also contains buttons for the commands to **Create** plot and to **Send Plot** to the plotter for the generation of a hard copy of the RI-1 diagrams.

- **Exit** command terminates ARI and gives control back to MicroStation for the generation of design files or for loading another application.

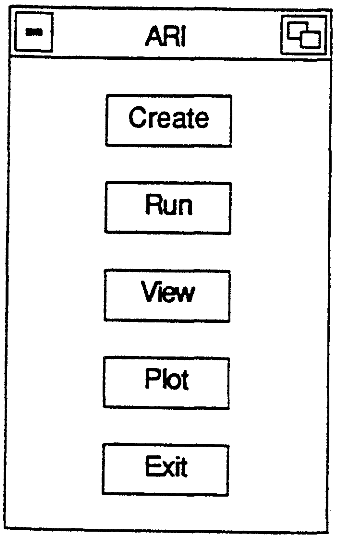


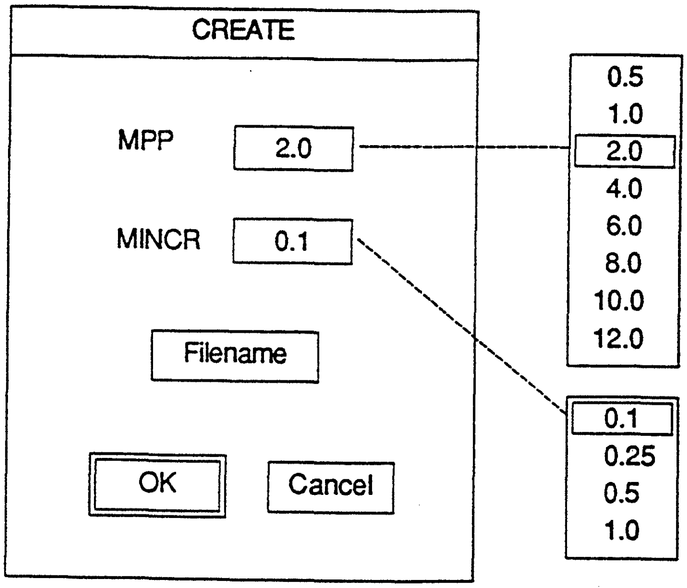
Figure 10. ARI Menu

2.4 CREATE

2.4.1 Create Command

When the Create command is selected from the ARI menu, a dialog box is brought up. A depiction of this box is shown in Figure 11. This dialog box allows to user to select the number of miles per page (MPP), the mile increment (MINCR), and the Filename.

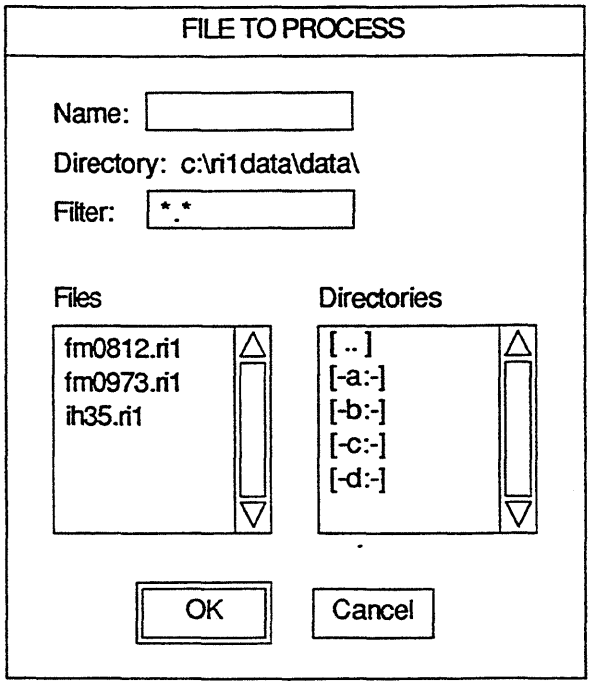
When the **Filename** command is selected from the options in the Create dialog box, a window for selecting the data file to be processed appears (Figure 12).



MPP 2.0

MINCR 0.1

Figure 11. Create Dialog Box



Name:

Directory: c:\ri1data\data\

Filter: *.*

Figure 12. File to Process Window

The following describes the numbers to be used for the Miles per Page, Mile Increment, and File to Load Data options on the create dialog box.

Select Miles per Page
- Default: 2.0
- Options: 0.5 1.0 2.0 4.0 6.0 8.0 10.0 12.0

Select Mile Increment
- Default: 0.1
- Options: 0.1 0.25 0.5 1.0

Select File to Load Data
- Default: None
- Options: (MicroStation file selection dialog box)

2.4.2 Run Command

Once the diagram parameters are selected (MPP, MINCR, and Filename), the **Run** command starts processing the data file. The ARI program makes two passes in each processing run. The first pass detects clutter in the display of the features, and the second pass creates and displays the diagrams on the screen. If there is clutter in the display of the roadway features, the **Clutter Detected** window comes up to let the user select **Automatic** or **Manual** processing of clutter.

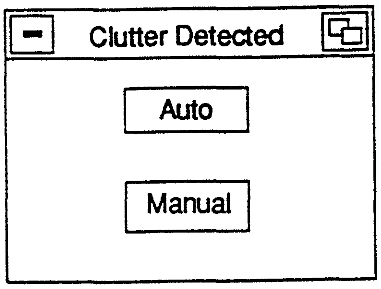
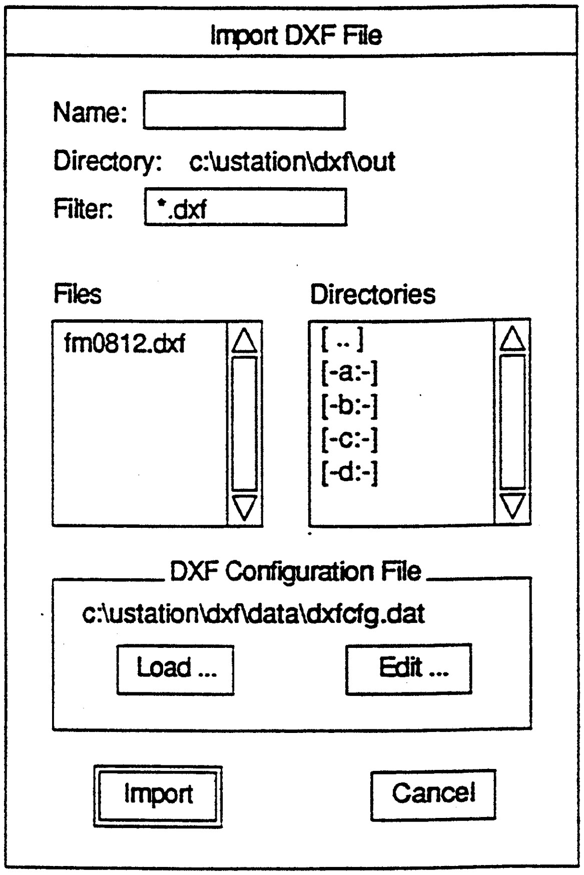


Figure 13. Clutter Detected Window

If **Auto** is selected, the optimal MPP which does not cause clutter is used for the generation of the diagrams. However, the smallest MPP that can be used is 0.5. This means that if there is still clutter at MPP=0.5, then clutter pages will be created in addition to the standard pages. The clutter pages resolve the clutter caused by overlapping graphic features. If **Manual** is selected, the user forces the use of the MPP selected in the **Create** window. This **Manual** selection will cause clutter if the selected MPP is larger than the optimal MPP calculated during the first pass of ARI. If clutter is detected and **Manual** processing of clutter is selected, then no clutter pages will be generated.

After the second ARI pass, a DXF file containing the graphical description of the diagrams is created. A dialog box titled **Import DXF File** (Figure 14) appears prompting the user to enter a filename to initiate the DXF to DGN conversion. At the end of this conversion, the diagrams are displayed on the screen, and the user can use the MicroStation View menu to preview the diagrams by zooming in/out, moving left/right or up/down.


Name:
Directory: c:\ustation\dxf\out
Filter: *.dxf

Files: fm0812.dxf

Directories: [..] [-a:-] [-b:-] [-c:-] [-d:-]

DXF Configuration File: c:\ustation\dxf\data\dxfcfg.dat

Figure 14. Import DXF File Dialog Box

2.5 VIEW DIAGRAMS COMMAND

The **View** command brings up a dialog box for the selection of the distance parameter displayed with each road feature in the diagram. The distance parameters that may be displayed are DFO, DIS, or DFODIS. The default is set at DFO. All three values are created with the diagram on three different MicroStation levels. After generation of the diagrams, and a view setting is selected

(DFO, DIS, or DFODIS), only the corresponding level is switched on. The other two levels are switched off. The View command dialog box is shown in Figure 15.

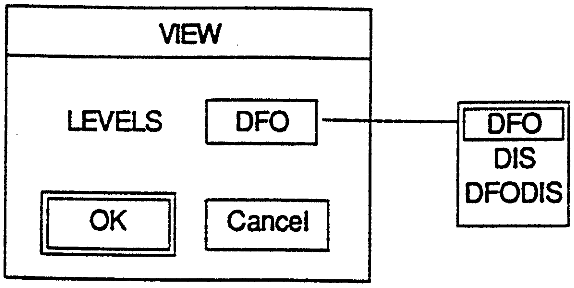


Figure 15. View Command Dialog Box

All MicroStation view commands (zoom in/out, move up/down and left/right) can be used at this point to preview the diagrams.

Select Location Display

- Default: DFO
- Options: DFO DIS Both

2.6 PLOT DIAGRAMS COMMAND

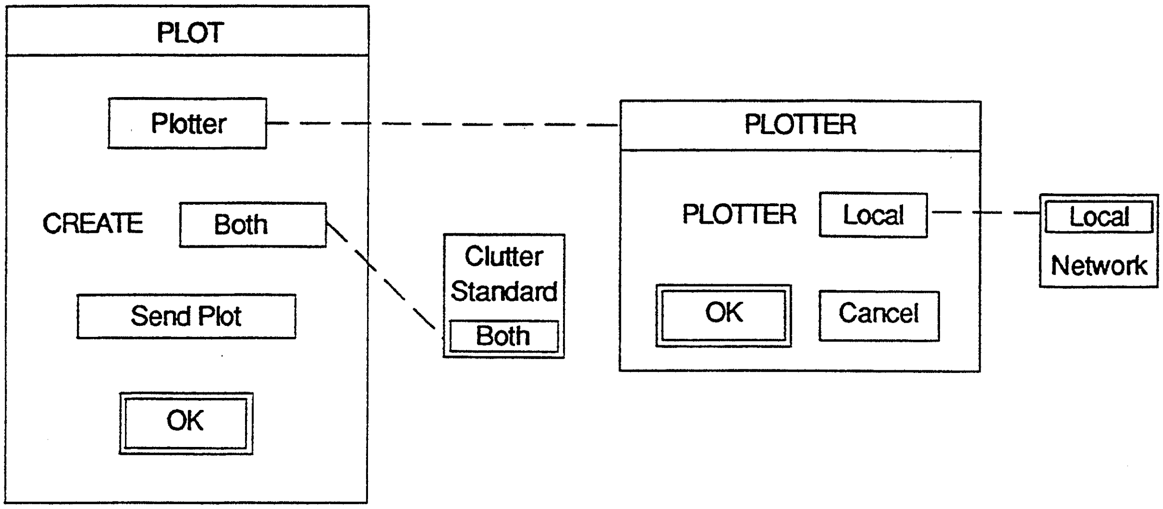
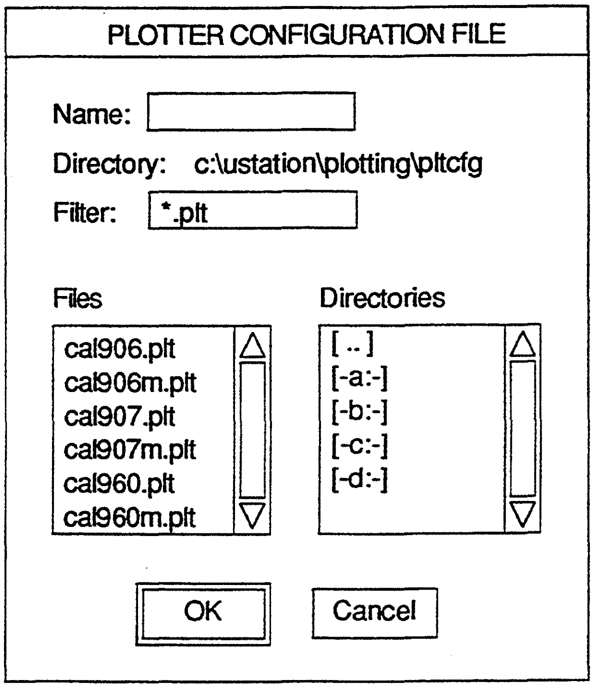


Figure 16. Plot Command Dialog Box

The **Plot** command brings up a dialog box (Figure 16) for the selection of the type of plotter: **Local** or **Network**. The Local plotter is the default plotter and is as specified in the MicroStation configuration file. If the RI-1 diagrams are already generated and displayed on the screen, then the **Create** plot command is to be selected followed by the **Send Plot** command for the generation of a hard copy of the diagrams. The **Create** plot command offers three options: **Clutter**, **Standard**, and **Both**. These options allow the user to select to plot the clutter diagrams only, the standard diagrams only, or both the standard and clutter diagrams.

When selecting the plotter to be used (**Local** or **Network**), a dialog box opens to allow the user to choose a local plotter or the queue name of the network plotter. If **Local** is selected, the following dialog box (Figure 17) opens to show all the local plotters:


PLOTTER CONFIGURATION FILE

Name:

Directory: c:\ustation\plotting\pltcfg

Filter: *.plt

Files
- cal906.plt
- cal906m.plt
- cal907.plt
- cal907m.plt
- cal960.plt
- cal960m.plt

Directories
- [..]
- [-a:-]
- [-b:-]
- [-c:-]
- [-d:-]

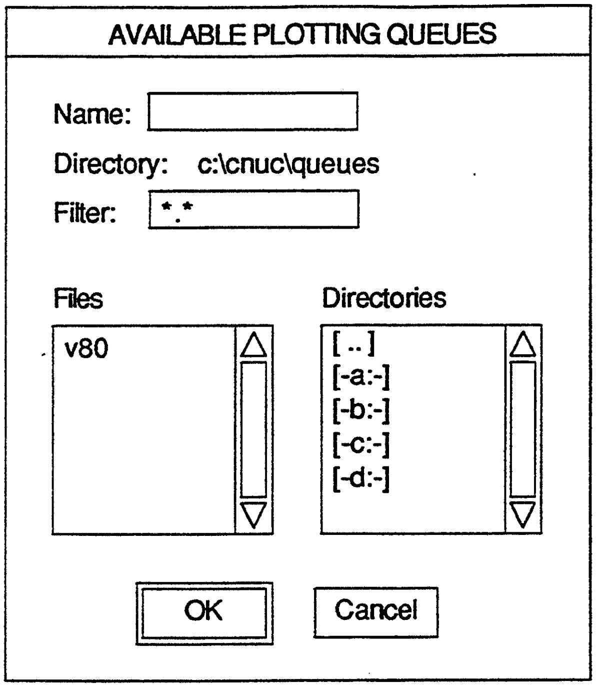
Figure 17. Local Plotter Dialog Box

If **Network** is selected, the dialog box depicted in Figure 18 opens to show all the queues available for network plotters.

The **Plot** command dialog box also allows the user to submit created diagrams to the plotter for the generation of a hard copy through use of the **Send Plot** button. Local plotting requires keyboard input to start plotting, as in most cases, the local plotter is a pen plotter and therefore uses single sheets that need to be fed individually. Network plotting is done through a batch process. For either plotter, the diagrams are plotted on B-size (17" by 11") sheets.

Select Plotter

- Default: Local
- Options: Local Network



AVAILABLE PLOTTING QUEUES

Name:

Directory: c:\cnuc\queues

Filter: *.*

Files

v80

Directories

[..]
[-a:-]
[-b:-]
[-c:-]
[-d:-]

Figure 18. Network Plotter Dialog Box

2.7 LOG OFF ARI

The **Exit** command is selected to unload ARI and give control back to MicroStation for the generation of design files or for loading another application.

When selecting some of ARI's commands, additional input may be required to complete a procedure. This will be indicated by the display of a prompt, submenu, and/or additional options. To respond to a prompt, the requested input can be typed or a selection made from a submenu. The mouse does not perform editing functions. Keys such as Del, Ins, Backspace, Home, End, left arrow, and right arrow can be used for editing.

3.0 EXAMPLE OUTPUT

After the initial phase of this project, the TRM team and the TTI research team developed a format for the automated RI-1 input data. The development of this format permitted TxDOT to begin work on data entry of the roadway feature inventory. In order for the TTI research team to test the ARI software, the TRM team provided TTI with test data files. The test data files were then processed and used to demonstrate ARI to the TxDOT staff. The test data files were also used in both the user and technical training conducted by TTI. This training was used to familiarize the TRM team with the ARI features. The user training showed users how to use ARI, its menus, and commands to generate a hard copy of RI-1 diagrams. The technical training aimed at teaching the programmers of the TRM team how to maintain the different software modules of ARI. The remainder of this chapter presents examples of input and output that represents a section of highway FM812.

3.1 THE INPUT: FILE FM0812.RI1

```
00 000.000 0 JT002 0 (02-13-92)
00 000.000 0 JT003 0 (FM0812)
00 000.000 0 JT004 0 (538+01.501)
00 000.000 0 JT005 0 (540+00.101)
00 000.000 0 JT006 0 (14)
00 000.000 0 JT007 0 (TRAVIS)
00 000.000 0 JT008 0 (STANDARD)
00 000.000 0 JT009 0 (NORMAL)
00 000.000 0 JT011 0 (DISPLACEMENT)
00 000.000 0 JT012 0 (ROUTE)
10 000.000 0 AT001 0 (STD)
10 000.000 0 AT004 0 (MAIN LANES)
10 000.000 0 AT007 0 (RI-1)
10 000.000 0 AT009 0 (FM 812)
10 000.000 0 AT010 0 (538+01.501)
10 000.000 0 AT011 0 (540+00.101)
10 000.000 0 AT012 0 (02-13-92)
10 000.000 0 BT001 0 (K)
10 000.000 K CT701 5 538 +01.501 001.570 (FT RB) (046) (BS) (FLEX) * *
10 000.000 K CT702 5 538 +01.501 001.570 (FT) (046) (SURF) (T61 ACP) * *
10 000.000 K CT703 5 538 +01.501 001.570 (LNS) (04) (DIR) (BOTH) * *
10 000.000 K CT704 5 538 +01.501 001.570 (FT) (00) (LT) (NO SH) * *
10 000.000 K CT705 5 538 +01.501 001.570 (FT) (00) (RT) (NO SH) * *
10 000.000 K CT706 5 538 +01.501 001.570 (LT) (NO CRB) (RT) (NO CRB) * *
10 000.000 K DT601 5 538 +01.501 001.570 (AXLOAD) (00) (TDLOAD) (00) (GRLOAD) (00)
10 000.000 K DT602 5 538 +01.501 001.570 (MXSPEED) (00) (MNSPEED) (00) * *
10 000.000 K ET601 5 538 +01.501 001.570 * (TWO-WAY X) (TRM COMP OPEN) * * *
10 000.000 K ET602 5 538 +01.501 001.570 (USROW) (100) (MNROW) (100) * *
10 000.000 K FT601 5 538 +01.501 001.570 (MAINT) (STATE) * * * *
10 000.000 K FT602 5 538 +01.501 001.570 (MDIST) (14) (MSECT) (10) * *
10 000.000 K GT601 5 538 +01.501 001.570 * (RURAL) (UA) (0) * *
10 000.000 K GT602 5 538 +01.501 001.570 (ADM) (01) (CN) (0) * *
10 000.000 K GT603 5 538 +01.501 001.570 * (R MAC) (FAS) * * *
10 000.000 K GT604 5 538 +01.501 001.570 * (ST HWY) (HPMS) (0) * *
10 000.000 K GT605 5 538 +01.501 001.570 * (NO SPL SYS) * * * *
10 000.000 K HT601 5 538 +01.501 001.570 (DIST) (14) (CO) (TRAVIS) (CS) (1149-01)
10 000.000 K HT602 5 538 +01.501 001.570 (CCR) (FM 973) * * * *
10 000.000 K HT603 5 538 +01.501 001.570 * (NO SEC RTE) * * * *
10 000.117 K BS214 3 538 +01.618 001.687 4100 (18"P) *
10 000.219 K BS112 3 538 +01.720 001.789 4000 (CR 1790) *
```

```
10 000.230 K BT410 1 538 +01.731 001.800 (S 26 45 00 W)
10 000.343 K CT711 5 538 +01.844 001.913 (FT RB) (026) * * * *
10 000.343 K CT712 5 538 +01.844 001.913 (FT) (024) * * * *
10 000.343 K CT713 5 538 +01.844 001.913 (LNS) (02) * * * *
10 000.343 K CT704 5 538 +01.844 001.913 (FT) (01) (LT) (GRVL SH) * *
10 000.343 K CT705 5 538 +01.844 001.913 (FT) (01) (RT) (GRVL SH) * *
10 000.431 K BS501 4 538 +01.932 002.001 (540) (002.001) * *
10 000.455 K BS402 4 540 +00.024 002.025 (=87 13 00 RT) (D=07 00 00 ) (L=0.236 T=0.148) *
10 000.532 K BS111 3 540 +00.101 002.102 2000 (FM 973) *
10 000.532 K XS999 2 540 +00.101 002.102
```

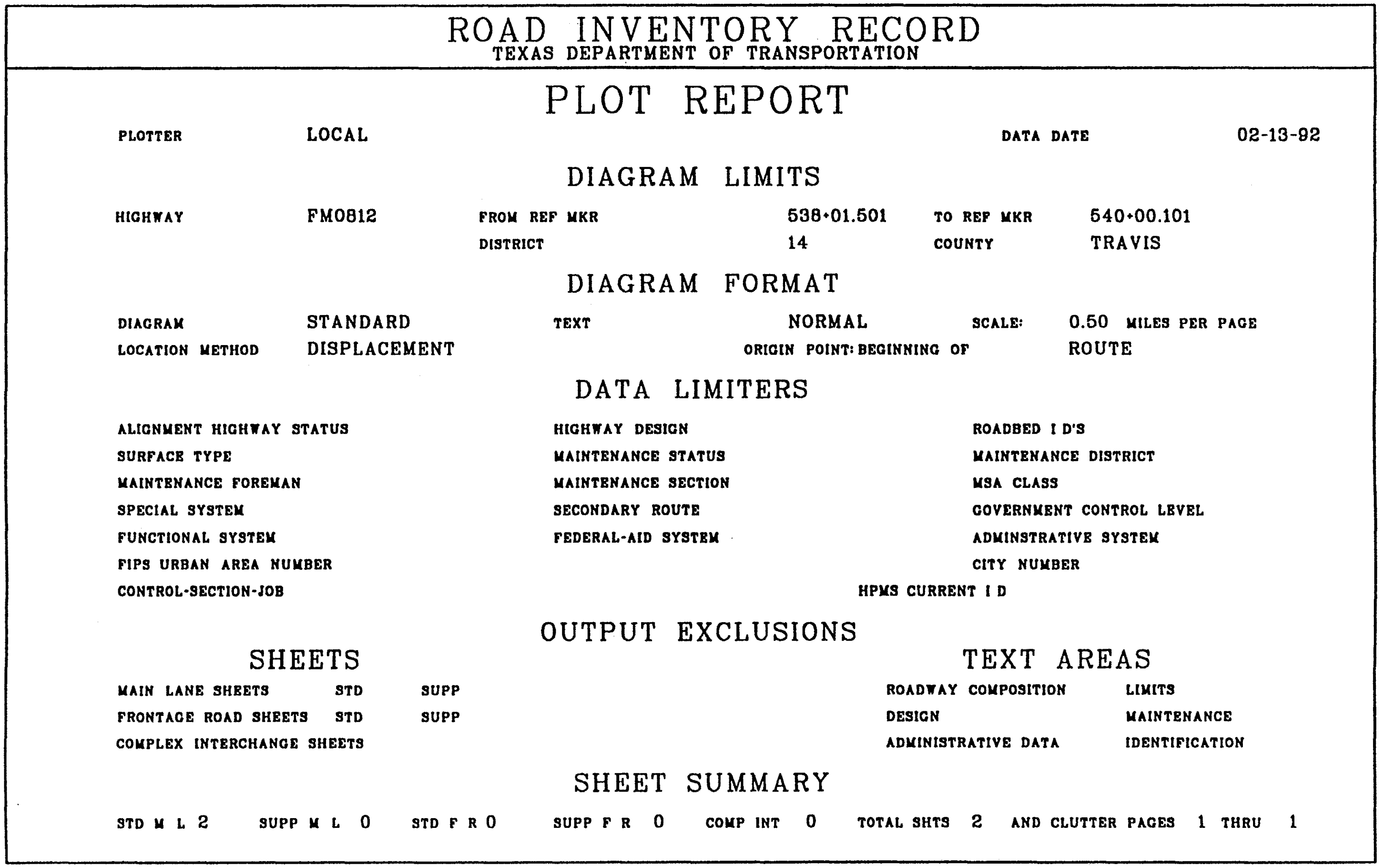
Note 1: * is a place holder for empty fields.
Note 2: Multi-word fields are between parentheses since the space character is used as a field separator.

3.2 THE OUTPUT: AUTOMATED RI-1 DIAGRAMS

For data file FM0812.RI1, the following drawings are generated:

- Plot report
- Two sheets of standard main lanes (K roadbed)
- One sheet of clutter resolution

Figures 19 through 22 show the output drawings of this section of highway FM812. It should be noted that the ARI diagrams are plotted on B-size (17" X 11") sheets; the following figures show the diagrams after photographic reduction.



ROAD INVENTORY RECORD
TEXAS DEPARTMENT OF TRANSPORTATION

PLOT REPORT

PLOTTER LOCAL DATA DATE 02-13-92

DIAGRAM LIMITS

HIGHWAY FM0812 FROM REF MKR 538+01.501 TO REF MKR 540+00.101
DISTRICT 14 COUNTY TRAVIS

DIAGRAM FORMAT

DIAGRAM STANDARD TEXT NORMAL SCALE: 0.50 MILES PER PAGE
LOCATION METHOD DISPLACEMENT ORIGIN POINT: BEGINNING OF ROUTE

DATA LIMITERS

ALIGNMENT HIGHWAY STATUS
SURFACE TYPE
MAINTENANCE FOREMAN
SPECIAL SYSTEM
FUNCTIONAL SYSTEM
FIPS URBAN AREA NUMBER
CONTROL-SECTION-JOB

HIGHWAY DESIGN
MAINTENANCE STATUS
MAINTENANCE SECTION
SECONDARY ROUTE
FEDERAL-AID SYSTEM

ROADBED I D'S
MAINTENANCE DISTRICT
MSA CLASS
GOVERNMENT CONTROL LEVEL
ADMINSTRATIVE SYSTEM
CITY NUMBER
HPMS CURRENT I D

OUTPUT EXCLUSIONS

SHEETS

MAIN LANE SHEETS STD SUPP
FRONTAGE ROAD SHEETS STD SUPP
COMPLEX INTERCHANGE SHEETS

TEXT AREAS

ROADWAY COMPOSITION LIMITS
DESIGN MAINTENANCE
ADMINISTRATIVE DATA IDENTIFICATION

SHEET SUMMARY

STD M L 2 SUPP M L 0 STD F R 0 SUPP F R 0 COMP INT 0 TOTAL SHTS 2 AND CLUTTER PAGES 1 THRU 1

Figure 19. Plot Report for the FM812 Test Data

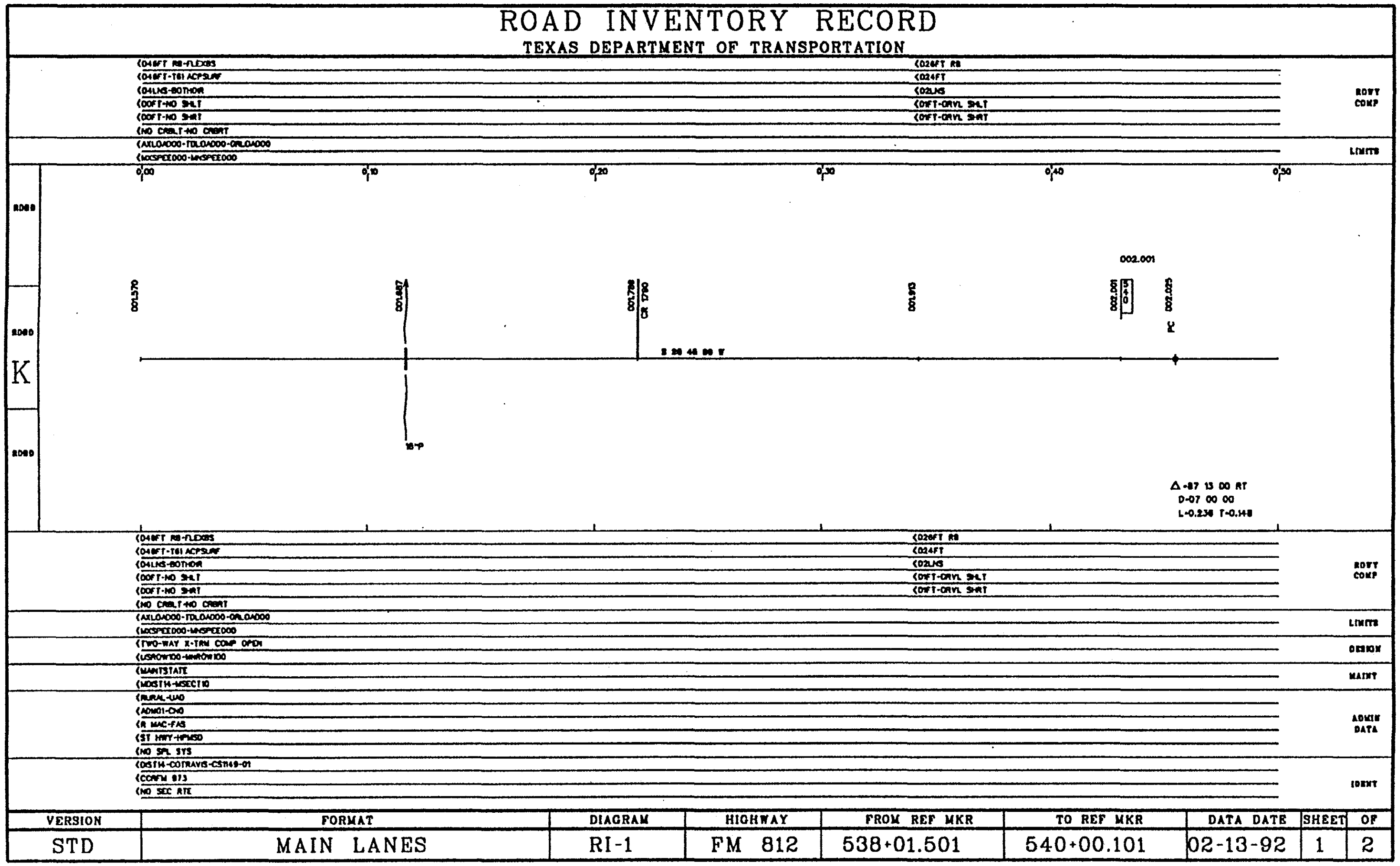
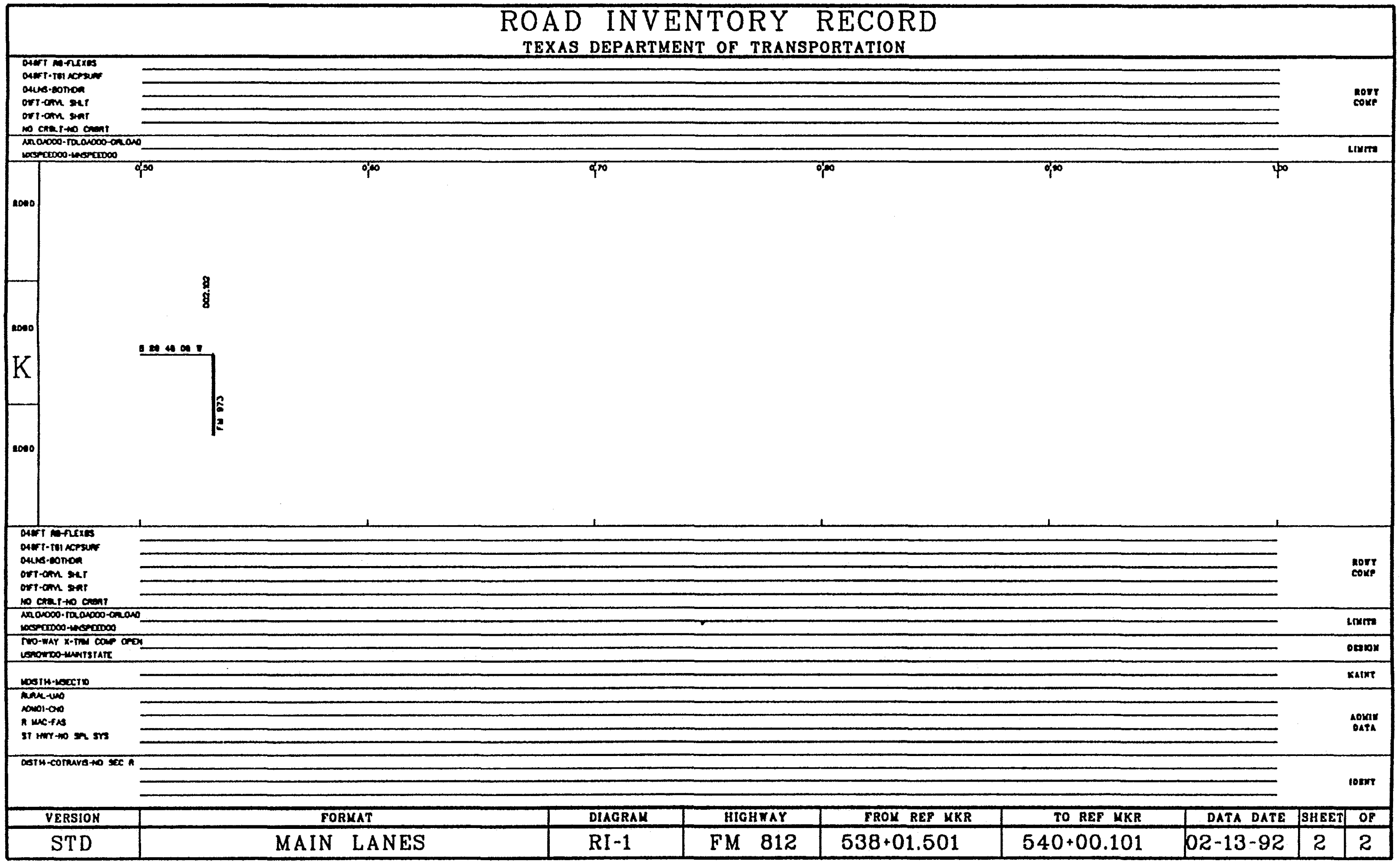


Figure 20. Sheet 1 of Standard Diagram Output for FM812 Test Data



ROAD INVENTORY RECORD

TEXAS DEPARTMENT OF TRANSPORTATION

04#FT AB-FLEXBS
04#FT-T01 ACPSURF
04LNS-BOTHDIR
01FT-GRVL SHLT
01FT-GRVL SHRT
NO CRBLT-NO CRBRT

AXLOAD00-TDLOAD00-GRLOAD
MXSPEED00-MNSPEED00

04#FT AB-FLEXBS
04#FT-T01 ACPSURF
04LNS-BOTHDIR
01FT-GRVL SHLT
01FT-GRVL SHRT
NO CRBLT-NO CRBRT

AXLOAD00-TDLOAD00-GRLOAD
MXSPEED00-MNSPEED00

TWO-WAY X-TRM COMP OPEN
USROW100-MAINTSTATE

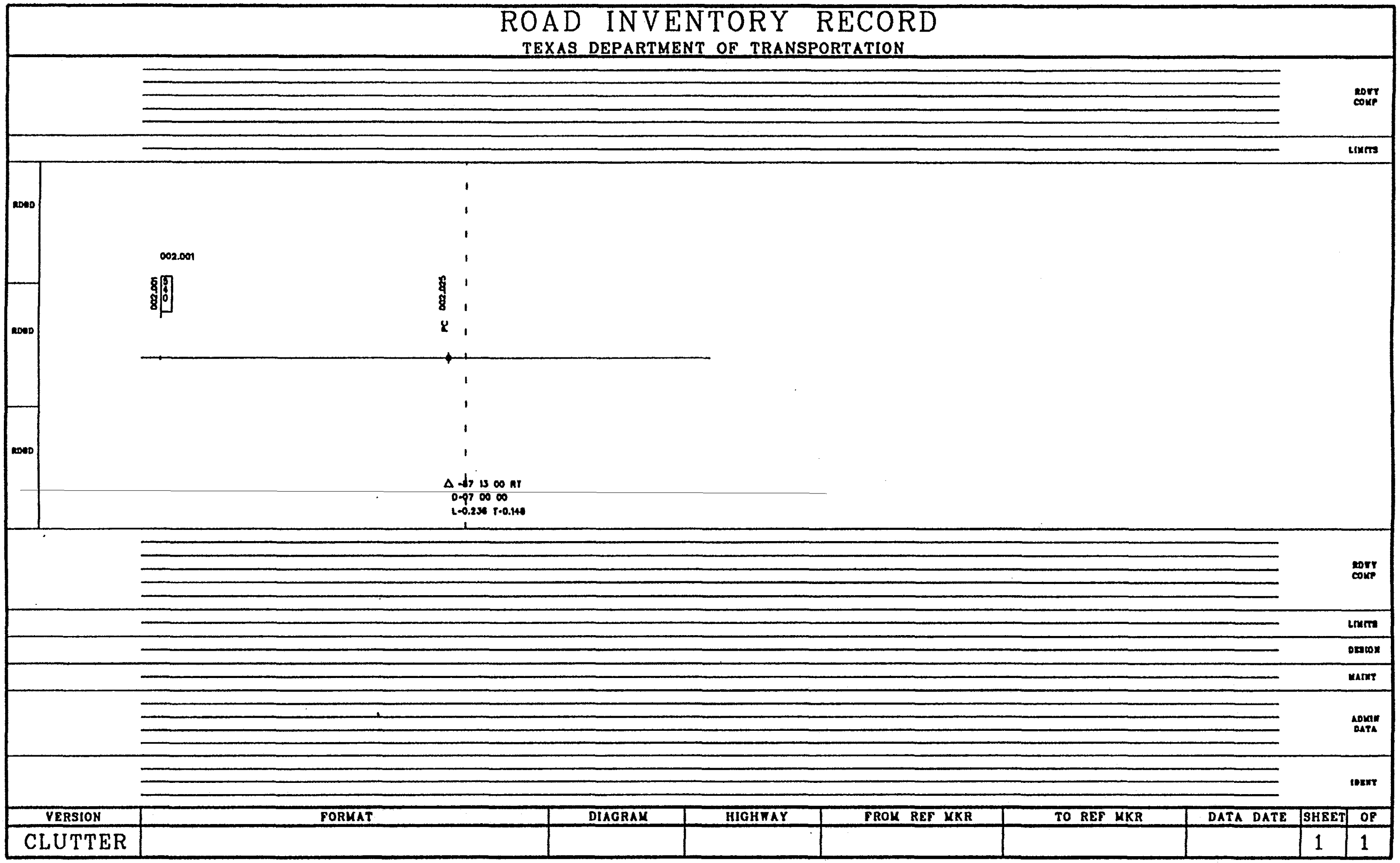
MDIST14-MSECT10

RURAL-UA0
ADM01-CH0
R MAC-FAS
ST HWY-NO SPL SYS

DIST14-COTRAVIS-NO SEC R

VERSION	FORMAT	DIAGRAM	HIGHWAY	FROM REF MKR	TO REF MKR	DATA DATE	SHEET	OF
STD	MAIN LANES	RI-1	FM 812	538+01.501	540+00.101	02-13-92	2	2

Figure 21. Sheet 2 of Standard Diagram Output for FM812 Test Data



ROAD INVENTORY RECORD

TEXAS DEPARTMENT OF TRANSPORTATION

VERSION	FORMAT	DIAGRAM	HIGHWAY	FROM REF MKR	TO REF MKR	DATA DATE	SHEET	OF
CLUTTER							1	1

Figure 22. Sheet 1 of Clutter Diagram Output for FM812 Test Data

4.0 DESCRIPTION OF PROGRAMS

Several programs were written to implement the different functions of ARI. Three computer languages were used so that full advantage was taken of the flexibility offered by the MicroStation 4.0 environment. The ARI diagram processing and the clutter analysis section of the code were written in C. The modules of the user interface were written in MDL and UCM. This chapter describes the different programs including the name of the file that contains the source code, the language in which they are written, their purpose, the format for their use, the arguments they need, the sub-programs they call, and the other programs from which they are called. A glossary of all the variables used in these programs is given in Appendix B.

4.1 ARI DIAGRAM PROCESSING

FILENAME : absymbol.c

MODULE :	function ab_features()	
LANGUAGE :	C	
PURPOSE :	To generate the required format for printing a string	
FORMAT :	ab_features(xoff,yoff,sheetno,a_symbolno,a_notation)	
RETURNS :	void	
ARG :	NAME	TYPE
	xoff	double
	yoff	double
	sheetno	int
	a_symbolno	char[6]
	a_notation	char[40]
CALLS :	get_info(),get_area_info(),Text()	
CALLED BY :	main(),input_processing(),draw_clutter_page()	

FILENAME : complex.c

MODULE :	function addcomplex()	
LANGUAGE :	C	
PURPOSE :	To create necessary DXF format for complex interchange symbol	
FORMAT :	addcomplex(hoff, voff, level, outfile, entfile, complex_name)	
RETURNS :	void	
ARG :	NAME	TYPE
	hoff	double
	voff	double
	level	char[3]

	outfile	FILE *
	entfile	FILE *
	complex_name	char[20]
CALLS :	InsBlk(),appendfile(),Text()	
CALLED BY :	main()	

FILENAME : drawcltr.c

MODULE :	function draw_clutter_page()	
LANGUAGE :	C	
PURPOSE :	To draw cluttered symbols with specific separation distance and vertical line at the end of each resolved clutter situation.	
FORMAT :	draw_clutter_page(cltrfile)	
RETURNS :	void	
ARG :	NAME	TYPE
	cltrfile	FILE *
CALLS :	drawpage_clutr(),draw_vertical_line(),draw_centerline()	
CALLED BY :	main()	

FILENAME : endhwy.c

MODULE :	function endhwy()	
LANGUAGE :	C	
PURPOSE :	To generate the required format for printing the end of highway string	
FORMAT :	endhwy (xoff,rdbd)	
RETURNS :	void	
ARG :	NAME	TYPE
	xoff	double
	rdbd	char
CALLS :	xs925()	
CALLED BY :	input_processing()	

FILENAME : fmap.c

MODULE :	function map_area_features()
LANGUAGE :	C
PURPOSE :	To build a table defining the relationship between areas of forms and features that go into these areas.
FORMAT :	map_area_features(feature,datafile)
RETURNS :	void

ARG :	NAME	TYPE
	feature	Feature_table
	datafile	char[40]
CALLS :		
CALLED BY :	main()	

FILENAME : formarea.c

MODULE :	function read_area_info()	
LANGUAGE :	C	
PURPOSE :	To build different tables from form0.dat, form1.dat and form2.dat	
FORMAT :	read_area_info(area, no_areas, datafile)	
RETURNS :	void	
ARG :	NAME	TYPE
	area[]	Area_Info
	no_areas	int
	datafile	char[40]
CALLS :	NONE	
CALLED BY :	main()	

FILENAME : formcnfg.c

MODULE :	function config_forms()
LANGUAGE :	C
PURPOSE :	To configure forms and different scale factors in MicroStation units
FORMAT :	config_forms()
RETURNS :	void
ARG :	NONE
CALLS :	NONE
CALLED BY :	main()

FILENAME : formdraw.c

MODULE :	drawform()	
LANGUAGE :	C	
PURPOSE :	To create DXF code to draw various forms	
FORMAT :	drawform(xoff, yoff, level, datafile, DxfFile)	
RETURNS :	void	
ARG :	NAME	TYPE
	xoff	double

	yoff	double
	level	char[3]
	datafile	char[80]
	DxfFile	FILE *
CALLS :	Line(),PolyLine(),Text(),Vertex(),Seqend()	
CALLED BY :	draw_new_page()	

FILENAME : formfile.c

MODULE :	function form_filename()	
LANGUAGE :	C	
PURPOSE :	To form an SDF filename depending upon rdbd	
FORMAT :	form_filename(input_rec,symfile)	
RETURNS :	symfile	
ARG :	NAME	TYPE
	input_rec	General_Input_Rec *
	symfile	char[20]
CALLS :	NONE	
CALLED BY :	input_processing(),draw_clutter_page()	

FILENAME : formkid.c

MODULE :	function read_no_of_child	
LANGUAGE :	C	
PURPOSE :	To form a table of composite symbols and their children	
FORMAT :	read_no_of_child(composite_symbols,no_symbols,datafile)	
RETURNS :	void	
ARG :	NAME	TYPE
	composite_symbols[]	Composite_Text_Symbol
	datafile	char[40]
	no_symbols	int *
CALLS :	NONE	
CALLED BY :	main()	

FILENAME : getarea.c

MODULE :	function get_area_info()
LANGUAGE :	C
PURPOSE :	To get the sheet area information
FORMAT :	get_area_info(feature, sheet_no, symbol_name, areas, count)

RETURNS : areas,count

ARG :

NAME	TYPE
feature	Feature_Table
sheet_no	int
symbol_name	char[6]
areas[]	Area_Info
count	int *

CALLS : get_info()

CALLED BY : ab_features(),txtprocess()

FILENAME : getchild.c

MODULE : function get_no_of_child()

LANGUAGE : C

PURPOSE : To find number of children of a composite symbol from a table

FORMAT : get_no_of_child(composite_symbols,symbolno,count1)

RETURNS : count1

ARG :

NAME	TYPE
composite_symbols[]	Composite_Text_Symbol
symbolno	char[6]
count1	int *

CALLS : NONE

CALLED BY : refresh_new_page(),txtprocess()

FILENAME : getcomp.c

MODULE : function get_composition()

LANGUAGE : C

PURPOSE : To get the composition (n+v,v+n) for text features

FORMAT : void get_composition(symbolno,field)

RETURNS : field

ARG :

NAME	TYPE
symbolno	char[10]
field	char[3]

CALLS : NONE

CALLED BY : txtprocess()

FILENAME : getinfo.c

MODULE : function get_info()

LANGUAGE : C
PURPOSE : To get the area information depending upon count
FORMAT : get_info(areainfo, no_areas, areanames, areas, count)
RETURNS : areas
ARG :

NAME	TYPE
areainfo[]	Area_Info
no_areas	int
areanames[]	char[6]
areas[]	Area_Info
count	int

CALLS : NONE
CALLED BY : ab_features(),draw_clutter_page(),endhwy(),getarea(),incrdraw(), input_processing(),main(),symboldraw(),getminmax()

FILENAME : getminmx.c

MODULE : function get_minmax()
LANGUAGE : C
PURPOSE : To find the minmax values (envelope) of a graphic symbol
FORMAT : get_minmax(sdffile, new_xmin,new_ymin,new_xmax,new_ymax)
RETURNS : new_xmin,new_ymin,new_xmax,new_ymax
ARG :

NAME	TYPE
sdffile	char[10]
new_xmin	double *
new_ymin	double *
new_xmax	double *
new_ymax	double *

CALLS : NONE
CALLED BY : draw_clutter_page()

FILENAME : getsym.c

MODULE : function get_symbol_value()
LANGUAGE : C
PURPOSE : To get the symbol value
FORMAT : get_symbol_value(simple_symbols,no_of_simple_textsym,symbolno,txtstr)
RETURNS : txtstr
ARG :

NAME	TYPE
simple_symbols[]	Simple_Text_Symbol
no_of_simple_textsym	int
symbolno	char[6]

		txtstr	char[40]
CALLS	:	NONE	
CALLED BY	:	refresh_new_page()	

FILENAME	**:**	**incrdraw.c**	
MODULE	:	function incrdraw()	
LANGUAGE	:	C	
PURPOSE	:	To draw mile increments	
FORMAT	:	incrdraw(xbegmile,xendmile,mileincr,begdraw,level,areaname,sheet,DxfFile)	
RETURNS	:	void	
ARG	:	NAME	TYPE
		xbegmile	double
		xendmile	double
		mileincr	double
		begdraw	double
		level	char[3]
		areaname[2]	char[6]
		sheet	int
		DxfFile	FILE *
CALLS	:	get_info(),Line(),Text()	
CALLED BY	:	draw_new_page()	

FILENAME	**:**	**inputprs.c**	
MODULE	:	function init()	
LANGUAGE	:	C	
PURPOSE	:	To initialize the input record	
FORMAT	:	init(input_rec)	
RETURNS	:	input_rec	
ARG	:	NAME	TYPE
		input_rec	General_Input_Rec *
CALLS	:	NONE	
CALLED BY	:	main()	

FILENAME	**:**	**inputprs.c**
MODULE	:	function fill_in_fields()
LANGUAGE	:	C
PURPOSE	:	To read the input record
FORMAT	:	fill_in_fields(input_rec, fp)

RETURNS :	input_rec	
ARG :	NAME	TYPE
	input_rec	General_Input_Rec *
	fp	FILE *
CALLS :	NONE	
CALLED BY :	main(),draw_clutter_page()	

FILENAME :	**inputprs.c**	
MODULE :	function input_processing()	
LANGUAGE :	C	
PURPOSE :	To process the input feature, record either text or graphic feature	
FORMAT :	input_processing(xoff, yoff, input_rec)	
RETURNS :	void	
ARG :	NAME	TYPE
	input_rec	General_Input_Rec *
	xoff	double
	yoff	double
CALLS :	ab_features(),j_features(),txtprocess(),endhwy(),form_filename(), get_info(),dfodis_placement(),symboldraw()	
CALLED BY :	main(),draw_clutter_page()	

FILENAME :	**inputprs.c**	
MODULE :	function dfodis_placement()	
LANGUAGE :	C	
PURPOSE :	To place dfo, dis for a feature	
FORMAT :	dfodis_placement(input_rec,voff)	
RETURNS :	void	
ARG :	NAME	TYPE
	input_rec	General_Input_Rec *
	voff	double
CALLS :	Text()	
CALLED BY :	input_processing()	

FILENAME :	**jsymbol.c**
MODULE :	function j_features()
LANGUAGE :	C
PURPOSE :	To generate the required format to print the strings of

the J features of the Plot Summary Report sheet
FORMAT : j_features(xoff, yoff, rec_info);
RETURNS : void
ARG :

NAME	TYPE
xoff	double
yoff	double
rec_info	General_Input_Rec

CALLS : Text()
CALLED BY : input_processing()

FILENAME : kidcomp.c

MODULE : function read_child_decomp()
LANGUAGE : C
PURPOSE : To build a table for normal and cryptic symbols with their decomposition and value
FORMAT : read_child_decomp(simple_symbols,no_symbols,datafile)
RETURNS : no_symbols
ARG :

NAME	TYPE
simple_symbols[]	Simple_Text_Symbol
no_symbols	int *
datafile	char[40]

CALLS : NONE
CALLED BY : main()

FILENAME : refresh.c

MODULE : function refresh_new_page()
LANGUAGE : C
PURPOSE : To refresh (place) text features on a new page
FORMAT : refresh_new_page(xbegform,ybegpage,sheetno)
RETURNS : void
ARG :

NAME	TYPE
xbegform	double
ybegpage	double
sheetno	int

CALLS : get_no_of_child(),get_symbol_value(),txtplace()
CALLED BY : draw_new_page()

FILENAME : symdraw.c

MODULE : function symboldraw()
LANGUAGE : C
PURPOSE : To create DXF code for a graphic symbol using its SDF
FORMAT : symboldraw(datafile,yoff,xoff,ybegpage,skew,level,rec_info,DxfFile)
RETURNS : void
ARG :

NAME	TYPE
DxfFile	FILE *
datafile	char[80]
level	char[3]
xoff	double
yoff	double
skew	double
ybegpage	double
rec_info	General_Input_Rec *

CALLS : Line(),Text(),Arc(),Circle(),PolyLine(),Vertex(),Sequend(),get_info()
CALLED BY : input_processing()

FILENAME : txtprs.c

MODULE : function txtprocess()
LANGUAGE : C
PURPOSE : To process a text feature and update its value
FORMAT : txtprocess(xoff,yoff,input_rec)
RETURNS : void
ARG :

NAME	TYPE
input_rec	General_Input_Rec *
xoff	double
yoff	double

CALLS : get_no_of_child(),get_composition(),update_symbol_value(),txtplace()
CALLED BY : input_processing()

FILENAME : txtplace.c

MODULE : function txtplace()
LANGUAGE : C
PURPOSE : To generate the DXF code for text
FORMAT : txtplace(xoff, yoff, txtstr1, sheetno, symbolname,rdbd)
RETURNS : void

ARG :

NAME	TYPE
xoff	double
yoff	double
txtstr1	char[80]
sheetno	int
symbolname	char[6]
rdbd	char

CALLS : xs924(),Text()
CALLED BY : refresh_new_page(),txtprocess()

FILENAME : update.c

MODULE : function update_symbol_value()
LANGUAGE : C
PURPOSE : To update symbol value
FORMAT : update_symbol_value(simple_symbols,no_of_simple_textsym,symbolno, txtstr)
RETURNS : void
ARG :

NAME	TYPE
simple_symbols[]	Simple_Text_Symbol
no_of_simple_textsym	int
symbolno	char[6]
txtstr	char[40]

CALLS : NONE
CALLED BY : txtprocess()

FILENAME : xs924

MODULE : function xs924()
LANGUAGE : C
PURPOSE : To generate the required format to draw the begin highway symbol
FORMAT : xs924(V_offset,Height);
RETURNS : void
ARG :

NAME	TYPE
V_offset	double
Height	double

CALLS : Line()
CALLED BY : txtplace()

FILENAME :	**xs925**	
MODULE :	function xs925()	
LANGUAGE :	C	
PURPOSE :	To generate the required format to draw the end highway symbol	
FORMAT :	xs925(V_offset,Height);	
RETURNS :	void	
ARG :	NAME	TYPE
	V_offset	double
	Height	double
CALLS :	Line()	
CALLED BY :	endhwy()	

FILENAME :	**sld.c**	
MODULE :	function draw_new_page()	
LANGUAGE :	C	
PURPOSE :	To draw new page	
FORMAT :	draw_new_page(sheet,new_sheet,rec_info)	
RETURNS :	void	
ARG :	NAME	TYPE
	sheet	int
	new_sheet	int
	rec_info	General_Input_Rec *
CALLS :	drawform(),incrdraw(),refresh_new_page(),ab_features()	
CALLED BY :	main()	

FILENAME :	**sld.c**	
MODULE :	function draw_centerline()	
LANGUAGE :	C	
PURPOSE :	To draw the roadbeds based upon the rdbd value	
FORMAT :	draw_centerline(rdbd_id,prev_xbegdraw,prev_enddraw,yoffset)	
RETURNS :	void	
ARG :	NAME	TYPE
	rdbd_id	char
	prev_xbegdraw	double
	prev_enddraw	double
	yoffset	double
CALLS :	PolyLine(),Vertex(),Sequend(),get_info()	
CALLED BY :	main(),draw_clutter_page()	

FILENAME :	**sld.c**	
MODULE :	function main()	
LANGUAGE :	C	
PURPOSE :	To create a DXF file from an ASCII input datafile	
FORMAT :	main()	
RETURNS :	void	
ARG :	NONE	
CALLS :	config_forms(),read_area_info(),read_no_of_child(),read_child_decomp(), map_area_features(),init(),fill_in_fields(),draw_new_page(),draw_centerline(),input_processing(),drawform(),addcomplex(),ab_features(),j_features(),get_info()	
CALLED BY :	NONE	

4.2 CLUTTER DETECTION AND PROCESSING

FILENAME :	**addsymcl.c**	
MODULE :	function add_symbol_clutter()	
LANGUAGE :	C	
PURPOSE :	To add the record of a symbol causing clutter to a file	
FORMAT :	add_symbol_clutter(rec_info,fclutter)	
RETURNS :	void	
ARG :	NAME	TYPE
	rec_info	General_Input_Rec *
	fclutter	FILE *
CALLS :	print_rec()	
CALLED BY :	main()	

FILENAME :	**addsymcl.c**
MODULE :	function print_rec()
LANGUAGE :	C
PURPOSE :	To produce the required format and write it to a file
FORMAT :	print_rec(rec_info, fp1)
RETURNS :	void

ARG :	NAME	TYPE
	rec_info	General_Input_Rec *
	fp1	FILE *
CALLS :	NONE	
CALLED BY :	add_symbol_clutter()	

FILENAME : bclutr.c

MODULE :	function b_clutter_txt()	
LANGUAGE :	C	
PURPOSE :	To detect the clutter in the text area or the graphic symbol area	
FORMAT :	b_clutter_txt(xoff, yoff, rec_info,no_of_notations,area_loc,fclutter)	
RETURNS :	void	
ARG :	NAME	TYPE
	xoff	double
	yoff	double
	rec_info	General_Input_Rec *
	no_of_notations	int
	area_loc	char[6]
	fclutter	FILE *
CALLS :	get_info()	
CALLED BY :	clutter()	

FILENAME : clutrset.c

MODULE :	function read_clutter_set()	
LANGUAGE :	C	
PURPOSE :	To build left and right boundary feature sets that can overlap	
FORMAT :	read_clutter_set(boundry_set,no_symbols,datafile)	
RETURNS :	boundry_set,no_symbols	
ARG :	NAME	TYPE
	boundry_set[]	char[6]
	datafile	char[40]
	no_symbols	int *
CALLS :	NONE	
CALLED BY :	main()	

FILENAME : clutrtxt.c

MODULE : function clutter_txt()
LANGUAGE : C
PURPOSE : To determine the clutter in text features and find optimal mpp
FORMAT : clutter_txt(xoff, yoff, txtstr1, rec_info,prev_text_rec,fclutter)
RETURNS : void
ARG :

NAME	TYPE
xoff	double
yoff	double
txtstr1	char[80]
rec_info	General_Input_Rec *
prev_text_rec	General_Input_Rec *
fclutter	FILE *

CALLS : get_area_info()
CALLED BY : txtprocess()

FILENAME : clutter.c

MODULE : function graphic_clutter()
LANGUAGE : C
PURPOSE : To determine the clutter in graphic feature and find optimal mpp
FORMAT : graphic_clutter(datafile,xoff,yoff, rec_info,prev_graphic_rec,fclutter)
RETURNS : void
ARG :

NAME	TYPE
datafile	char[80]
xoff	double
yoff	double
rec_info	General_Input_Rec *
prev_graphic_rec	General_Input_Rec *
fclutter	FILE *

CALLS : get_minmax(),get_info(),get_symbol_set(),add_symbol_clutter(),b_clutter_txt()
CALLED BY : input_processing()

FILENAME : fmap.c

MODULE : map_area_features()
LANGUAGE : C
PURPOSE : To build a table defining relationship between areas of forms and features that go into these areas.

FORMAT : map_area_features(feature,datafile)
RETURNS : void
ARG :

NAME	TYPE
feature	Feature_table
datafile	char[40]

CALLS : NONE
CALLED BY : main()

FILENAME : formarea.c

MODULE : function read_area_info()
LANGUAGE : C
PURPOSE : To build different tables from form0.dat,form1.dat,form2.dat
FORMAT : read_area_info(area, no_areas, datafile)
RETURNS : void
ARG :

NAME	TYPE
area[]	Area_Info
no_areas	int
datafile	char[40]

CALLS : NONE
CALLED BY : main()

FILENAME : formcnfg.c

MODULE : function config_forms()
LANGUAGE : C
PURPOSE : To configure forms and different scales in MicroStation units
FORMAT : config_forms()
ARG : NONE
CALLS : NONE
CALLED BY : main()

FILENAME : formfile.c

MODULE : function form_filename()
LANGUAGE : C
PURPOSE : To form an SDF filename based upon rdbd value
FORMAT : form_filename(input_rec,symfile)
RETURNS : symfile

ARG	:	NAME	TYPE
		input_rec	General_Input_Rec *
		symfile	char[20]
CALLS	:	NONE	
CALLED BY	:	input_processing()	

FILENAME : formkid.c

MODULE	:	function read_no_of_child()	
LANGUAGE	:	C	
PURPOSE	:	To form a table of composite symbols and their children	
FORMAT	:	read_no_of_child(composite_symbols,no_symbols,datafile)	
RETURNS	:	void	
ARG	:	NAME	TYPE
		composite_symbols[]	Composite_Text_Symbol
		datafile	char[40]
		no_symbols	int *
CALLS	:	NONE	
CALLED BY	:	main()	

FILENAME : getarea.c

MODULE	:	function get_area_info()	
LANGUAGE	:	C	
PURPOSE	:	To get the area information	
FORMAT	:	get_area_info(feature, sheet_no, symbol_name, areas,count)	
RETURNS	:	areas,count	
ARG	:	NAME	TYPE
		feature	Feature_Table
		sheet_no	int
		symbol_name	char[6]
		areas[]	Area_Info
		count	int *
CALLS	:	get_info()	
CALLED BY	:	clutter_txt()	

FILENAME : getchild.c

MODULE	:	function get_no_of_child()

LANGUAGE : C
PURPOSE : To find number of children of a composite symbol from a table
FORMAT : get_no_of_child(composite_symbols,symbolno,count1)
RETURNS : count1
ARG :

NAME	TYPE
composite_symbols[]	Composite_Text_Symbol
symbolno	char[6]
count1	int *

CALLS : NONE
CALLED BY : txtprocess()

FILENAME : getcomp.c

MODULE : function get_composition()
LANGUAGE : C
PURPOSE : To get the composition(n+v,v+n) for text features
FORMAT : get_composition(symbolno,field)
RETURNS : field
ARG :

NAME	TYPE
symbolno	char[10]
field	char[3]

CALLS : NONE
CALLED BY : txtprocess()

FILENAME : getinfo.c

MODULE : function get_info()
LANGUAGE : C
PURPOSE : To get the area information based upon count
FORMAT : get_info(areainfo, no_areas, areanames, areas, count)
RETURNS : areas
ARG :

NAME	TYPE
areainfo[]	Area_Info
no_areas	int
areanames[]	char[6]
areas[]	Area_Info
count	int

CALLS : NONE
CALLED BY : b_clutter_txt(),graphic_clutter(),getarea(),

FILENAME : getminmx.c

MODULE : function get_minmax()
LANGUAGE : C
PURPOSE : To find the minmax values (envelope) of a graphic symbol
FORMAT : get_minmax(sdffile,new_xmin,new_ymin,new_xmax,new_ymax)
RETURNS : new_xmin,new_ymin,new_xmax,new_ymax
ARG :

NAME	TYPE
sdffile	char[10]
new_xmin	double *
new_ymin	double *
new_xmax	double *
new_ymax	double *

CALLS : NONE
CALLED BY : graphic_clutter()

FILENAME : getset.c

MODULE : function get_symbol_set()
LANGUAGE : C
PURPOSE : To find if a graphic feature belongs to the boundary feature sets
FORMAT : int get_symbol_set(boundry_set,no_of_set,symbolno,setno)
RETURNS :
ARG :

NAME	TYPE
boundry_set[]	char[6]
no_of_set	int
setno	int
symbolno	char[6]

CALLS : NONE
CALLED BY : graphic_clutter()

FILENAME : inputprs.c

MODULE : function init()
LANGUAGE : C
PURPOSE : To initialize the input record
FORMAT : init(input_rec)
RETURNS : input_rec
ARG :

NAME	TYPE
input_rec	General_Input_Rec *

CALLS : NONE
CALLED BY : main()

FILENAME : inputprs.c

MODULE : function fill_in_fields()
LANGUAGE : C
PURPOSE : To read the input record
FORMAT : fill_in_fields(input_rec, fp)
RETURNS : input_rec
ARG :

NAME	TYPE
input_rec	General_Input_Rec *
fp	FILE *

CALLS : NONE
CALLED BY : main()

FILENAME : inputprs.c

MODULE : function input_processing
LANGUAGE : C
PURPOSE : To process the input feature record either text or graphic feature
FORMAT : input_processing(xoff, yoff, input_rec,prev_input_rec,fclutter)
RETURNS : void
ARG :

NAME	TYPE
xoff	double
yoff	double
input_rec	General_Input_Rec *
prev_input_rec	General_Input_Rec *
fclutter	FILE *

CALLS : txtprocess(),form_filename(),graphic_clutter()
CALLED BY : main()

FILENAME : kidcomp.c

MODULE : function read_child_decomp()
LANGUAGE : C
PURPOSE : To build a table for normal and cryptic symbols with their decomposition and value
FORMAT : read_child_decomp(simple_symbols,no_symbols,datafile)

RETURNS :	no_symbols	
ARG :	NAME	TYPE
	simple_symbols[]	Simple_Text_Symbol
	no_symbols	int *
	datafile	char[40]
CALLS :	NONE	
CALLED BY :	main()	

FILENAME : txtprs.c

MODULE :	function txtprocess()	
LANGUAGE :	C	
PURPOSE :	To process a text feature and update its value	
FORMAT :	txtprocess(xoff,yoff,input_rec,prev_text_rec,fclutter)	
RETURNS :	void	
ARG :	NAME	TYPE
	xoff	double
	yoff	double
	input_rec	General_Input_Rec *
	prev_text_rec	General_Input_Rec *
	fclutter	FILE *
CALLS :	get_no_of_child(),get_composition(),update_symbol_value(),clutter_txt()	
CALLED BY :	input_processing()	

FILENAME : update.c

MODULE :	function update_symbol_value()	
LANGUAGE :	C	
PURPOSE :	To update symbol value	
FORMAT :	update_symbol_value(simple_symbols,no_of_simple_textsym,symbolno,txtstr)	
RETURNS :	void	
ARG :	NAME	TYPE
	simple_symbols[]	Simple_Text_Symbol
	no_of_simple_textsym	int
	symbolno	char[6]
	txtstr	char[40]
CALLS :	NONE	
CALLED BY :	txtprocess()	

FILENAME :	**clutrari.c**	
MODULE :	function main()	
LANGUAGE :	C	
PURPOSE :	To detect clutter in datafile and find optimal mpp	
FORMAT :	main()	
RETURNS :	void	
ARG :	NONE	
CALLS :	config_forms(),read_area_info(),read_no_of_child(),read_child_decomp(), read_clutter_set(),map_area_features(),init(),fill_in_fields(), fill_prev_rec(),draw_new_page(),input_processing()	
CALLED BY :	NONE	

FILENAME :	**clutrari.c**	
MODULE :	function draw_new_page()	
LANGUAGE :	C	
PURPOSE :	To calculate the MicroStation values for a new page	
FORMAT :	draw_new_page(sheet)	
RETURNS :	void	
ARG :	NAME	TYPE
	sheet	int
CALLS :	NONE	
CALLED BY :	main()	

4.3 USER INTERFACE

FILENAME :	**ari.mc**	
MODULE :	function main()	
LANGUAGE :	MicroStation Development Language	
PURPOSE :	To drive the MDL user interface.	
FORMAT :	main (int argc, char *argv[])	
RETURNS :	void	
ARG :	NONE	

FILENAME : ari.mc

MODULE : function basic_myok()
LANGUAGE : MicroStation Development Language
PURPOSE : Hook function for OK button.
FORMAT : Private void basic_myok
(DialogItemMessage *dimP)
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function basic_pagestatus()
LANGUAGE : MicroStation Development Language
PURPOSE : Hook function to read the user option to plot pages (standard, clutter or both).
FORMAT : Private void basic_pagestatus
(DialogItemMessage *dimP)
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function select_plot()
LANGUAGE : MicroStation Development Language
PURPOSE : Hook function to read the user option for plotter (local or network). Also, to open a dialog box to select the actual local plotter name or network queuename.
FORMAT : Private void select_plot
(DialogItemMessage *dimP)
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function basic_exit()
LANGUAGE : MicroStation Development Language
PURPOSE : To quit ari.ma MDL application.
FORMAT : Private void basic_exit
(DialogItemMessage *dimP)

RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function basic_run()
LANGUAGE : MicroStation Development Language
PURPOSE : To start running the ari input datafile processing, open clutter dialog box if clutter is detected, else run the program to start creating diagrams.
FORMAT : Private void basic_run
(DialogItemMessage *dimP)
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function basic_runpart2()
LANGUAGE : MicroStation Development Language
PURPOSE : To run the C program to generate the DXF file, convert DXF to DGN, and execute the UCM to display distance from origin (DFO) as default.
FORMAT : Private void basic_runpart2()
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function basic_create_standard_plot()
LANGUAGE : MicroStation Development Language
PURPOSE : To execute a UCM to fence each diagram and produce a plot file for local plotter. Also create an ASCII file containing a list of plot filenames.
FORMAT : Private void basic_create_standard_plot(nopagetemp,
filenametemp5,xlowtemp,ylowtemp,pagelength,pagewidth,
pagedistance,command2temp)
RETURNS : void

ARG :

NAME	TYPE
nopagetemp	int
filenametemp5	char[120]
command2temp	char[120]
xlowtemp	double
ylowtemp	double
pagelength	double
pagewidth	double
pagedistance	double

FILENAME : ari.mc

MODULE : function basic_create_clutter_plot()

LANGUAGE : MicroStation Development Language

PURPOSE : To execute a UCM to fence each diagram of clutter pages and produce a plot file for local plotter. Also, to create an ASCII file containing a list of plot filenames.

FORMAT : Private void basic_create_clutter_plot(nopagetemp,clutterflag, filenametemp3,xlowtemp1,ylowtemp1,pagelength,pagewidth,pagedistance, command2temp)

RETURNS : void

ARG :

NAME	TYPE
nopagetemp	int
clutterflag	int
filenametemp3	char[120]
command2temp	char[120]
xlowtemp1	double
ylowtemp1	double
pagelength	double
pagewidth	double
pagedistance	double

FILENAME : ari.mc

MODULE : function basic_iplot_create()

LANGUAGE : MicroStation Development Language

PURPOSE : To create a file with commands to plot on a network plotter. Also, to execute a UCM to produce a fence around each page and put its contents in a DGN file.

FORMAT : basic_iplot_create(no_pages,fname,fp,xbegfence,ybegfence,page_length, page_width,sep_distance)

RETURNS : void

ARG :

NAME	TYPE
no_pages	int
fname	char[20]
fp	FILE *
xbegfence	double
ybegfence	double
page_length	double
page_width	double
sep_distance	double

FILENAME : ari.mc

MODULE : function basic_create_plot()
LANGUAGE : MicroStation Development Language
PURPOSE : To call the appropriate functions based upon the plotter selection and the choice of which sheets to plot.
FORMAT : Private void basic_create_plot()
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function send_plot()
LANGUAGE : MicroStation Development Language
PURPOSE : To send created plotfiles to local or network plotter.
FORMAT : Private void send_plot()
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : command basic_openModal1()
LANGUAGE : MicroStation Development Language
PURPOSE : To open a dialog box for mpp, mincr and input data filename.
FORMAT : Public cmdName void basic_openModal1
(char *unparsedP) cmdNumber CMD_OPENMODAL1
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : command basic_openModal2()
LANGUAGE : MicroStation Development Language
PURPOSE : To execute UCMs to show DFO, DIS or both.
FORMAT : Public cmdName void basic_openModal2
(char *unparsedP) cmdNumber CMD_OPENMODAL2
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : command basic_auto()
LANGUAGE : MicroStation Development Language
PURPOSE : To run the program for creation of diagrams in DXF format with optimal mpp.
FORMAT : Public cmdName void basic_auto
(char *unparsedP) cmdNumber CMD_auto
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : command basic_manual()
LANGUAGE : MicroStation Development Language
PURPOSE : To run the program for creation of diagrams in DXF format with user selected mpp.
FORMAT : Public cmdName void basic_manual
(char *unparsedP) cmdNumber CMD_manual
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function listfile_getfile()
LANGUAGE : MicroStation Development Language
PURPOSE : To list filenames of input data files for the user to select from.
FORMAT : Private boolean listfile_getfile /* <= TRUE if error */
(char *tempFileName)

RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function listfile_plotterfile()
LANGUAGE : MicroStation Development Language
PURPOSE : To list filenames of available local plotters for the user to select from.
FORMAT : Private boolean listfile_plotterfile /* <= TRUE if error */
(char *plotFileName)
RETURNS : void
ARG : NONE

FILENAME : ari.mc

MODULE : function listfile_queuefile()
LANGUAGE : MicroStation Development Language
PURPOSE : To list available network plotter queues for the user to select from.
FORMAT : Private boolean listfile_queuefile /* <= TRUE if error */
(char *queueFileName)
RETURNS : void
ARG : NONE

FILENAME : dfo.ucm

MODULE :
LANGUAGE : User Command Macro
PURPOSE : To display distance from origin only by turning on MicroStation level 5 and turning off levels 4,6.
FORMAT :
RETURNS :
ARG : NONE

FILENAME : dis.ucm

MODULE :
LANGUAGE : User Command Macro
PURPOSE : To display displacement only by turning on MicroStation level 6 and turning off levels 4,5.

FORMAT :
RETURNS :
ARG : NONE

FILENAME : dfodis.ucm

MODULE :
LANGUAGE : User Command Macro
PURPOSE : To display distance from origin and displacement by turning on MicroStation level 4 and turning off levels 5,6.
FORMAT :
RETURNS :
ARG : NONE

FILENAME : netplt.ucm

MODULE :
LANGUAGE : User Command Macro
PURPOSE : To place a fence around each page based upon the number of standard and/or clutter pages and create a DGN file for plotting.
FORMAT :
RETURNS :
ARG : x_coordinate,y_coordinate,page_length,page_width, page_separation_distance,no_of_pages

---

FILENAME : localplt.ucm

MODULE :
LANGUAGE : User Command Macro
PURPOSE : To place a fence around each page based upon the number of standard and/or clutter pages and create a plotfile for plotting.
FORMAT :
RETURNS :
ARG : x_coordinate,y_coordinate,page_length,page_width, page_separation_distance,no_of_pages

---

5.0 CONCLUSIONS

ARI was developed as a software application that runs within the MicroStation 4.0 environment. ARI provides full automation to generate a hard copy of RI-1 diagrams from an ASCII input data file that contains a roadway features description. The use of MicroStation 4.0 as the graphic environment provides a friendly and easy graphic user interface. Several clear and simple menus allow the user to select from the different available options, create RI-1 diagrams, preview them, and send them to either a local or network plotter.

The specifications provided by TxDOT for the development of ARI were mainly hand drawings of the desired output. In order to meet these specifications, TTI has developed the Symbol Description Format (SDF) which is an ASCII description of graphic symbols in their elementary drawing components. SDF gives ARI great built-in flexibility and makes it very easy to customize. This customization ability opens the door to a multitude of other applications that require graphical representation of textual database information.

The ARI software application is a very useful tool, not only for the districts' users but also for the TRM team. The districts' users can use ARI to automatically generate a hard copy of RI-1 diagrams instead of relying on the old hand drawn version, and TRM can benefit from ARI by using it to verify the RI-1 database and to maintain its integrity.

6.0 APPENDIX A

SYMBOL DESCRIPTION FORMAT (SDF)

6.1 LINE

L linestyle linetype beginwidth endwidth X1 Y1 X2 Y2

Variable Name	Description	Variable Type
L	Line	char
linestyle	Regular (R/r) or Polyline (P/p)	char
linetype	Continuous (C/c) or Dotted (D/d)	char
beginwidth	Beginning width of line	double
endwidth	Ending width of line (Begin and end width define line weight -- line wt. betw. 0.0 and 9.0)	double
X1, Y1	Start coordinates of line in in.	double
X2, Y2	End coordinates of line in in.	double

Example: L P C 5.0 5.0 0.0 0.0 0.0 0.625
Continuous polyline with starting and ending width 5.0, starting at (0.0,0.0) and ending at (0.0,0.625).

6.2 CIRCLE

C linetype xcenter ycenter radius linewidth

Variable Name	Description	Variable Type
C	Circle	char
linetype	Continuous (C/c) or dotted (D/d)	char
xcenter,ycenter	Center coordinates	double
radius	Circle radius in in.	double
linewidth	Line weight (between 0.0 and 9.0)	double

Example: C D 0.0 0.0 3.0 2.0
Dotted circle with center at origin (0.0,0.0), radius equal to 3.0 in. and line weight 2.0.

6.3 ARC

A linetype linestyle xcenter ycenter radius startangle endangle linewidth

Variable Name	Description	Variable Type
A	Arc	char
linetype	Continuous (C/c) or dotted (D/d)	char
linestyle	Regular (R/r) or Polyline (P/p)	char
xcenter,ycenter	Center of arc	double
radius	Radius of arc in in.	double
startangle	Start angle of arc	double
endangle	End angle of arc (Start and end angles are specified anticlockwise)	double
linewidth	Line weight (between 0.0 and 9.0)	double

Example: A D P 0.0 0.0 2.0 30 90 1.0
Dotted arc with center at (0.0,0.0), angle 60 and line weight 1.0.

6.4 TEXT (PRESENT IN GRAPHIC SYMBOL)

T position angle XLL YLL XUR YUR textstyle chsize chwidth text

Variable Name	Description	Variable Type
T	Text	char
position	Text position in a box C or c = centered L or l = left justified R or r = right justified	char
XLL, YLL	Lower left coordinates	double
XUR, YUR	Upper right coordinates	double
textstyle	Text style as defined in uStation; e.g., ROMAN	string
chsize	Character size	double
chwidth	Character width	double
Text	Actual text present in symbol	string

Example: T C 0.0 3.0 0.0 6.0 3.0 ROMAN 3.0 0.0 BOUNDARY
The text string "BOUNDARY" is centered in a box defined by [(3.0,0.0),(6.0,3.0)] with character size 3.0 and font type Roman.

6.5 NOTATION TEXT (PRESENT IN THE DATA FILE)

N angle hoff area textstyle chsize chwidth

Variable Name	Description	Variable Type
N	Notation text	char
hoff	Horizontal offset	double
angle	Angle of text	double
area	Area on form where text is going to be placed	string
textstyle	Text style	double
chsize	Character size	double
chwidth	Character width	double

Example: N 0.0 0.0 b4 ROMAN 3.0 0.0
Text with graphic symbol in form area b4 with character size 3.0 and font type Roman.

6.6 VALUE TEXT (TEXT IN THE DATA FILE AT ANGLE = 90.0)

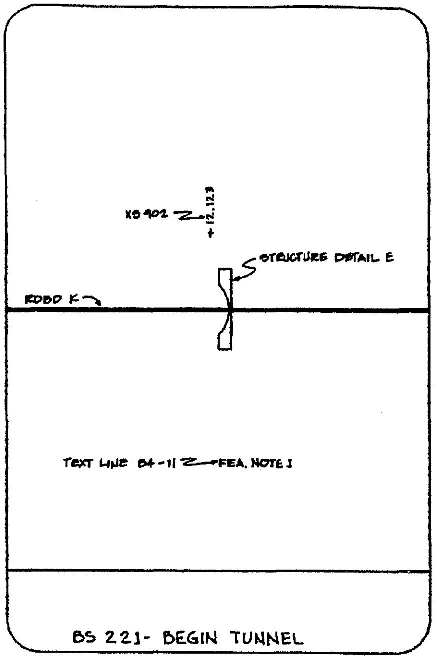
V position angle XLL YLL XUR YUR textstyle chsize chwidth

Variable Name	Description	Variable Type
V	Value text	char
position	Text position in a box C or c = centered L or l = left justified R or r = right justified	char
angle	Angle of text	double
XLL, YLL	Lower left coordinates	double
XUR, YUR	Upper right coordinates	double
textstyle	Text style	double
chsize	Character size	double
chwidth	Character width	double

Example: V C 90.0 3.0 0.0 3.0 0.0 ROMAN 3.0 0.0
Vertical text at angle 90.0 found in data file with character size 3.0 and Roman font type.

The following two examples show how graphic symbols are described in SDF. The specifications give the hand drawing of the symbols as well as their dimensions. The two examples selected are structure symbol BS221 - Begin Tunnel, and boundary symbol BS321 - Park Exit.

Example 1: BS221 - Begin Tunnel



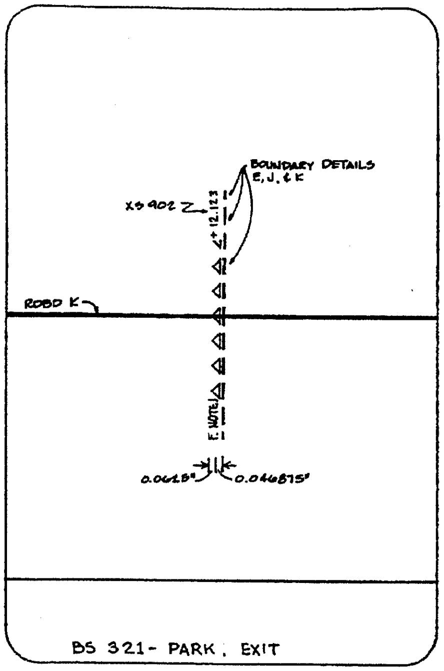
File: BS221KC.SDF

```
L P C    2.0    2.0    0.1015625    0.0        0.1015625    0.34375
L P C    2.0    2.0    0.0          0.21875    0.0          0.34375
L P C    2.0    2.0    0.0          0.34375    0.1015625    0.34375
L P C    2.0    2.0    0.1015625    0.0        0.1015625   -0.34375
L P C    2.0    2.0    0.0         -0.21875    0.0         -0.34375
L P C    2.0    2.0    0.0         -0.34375    0.1015625   -0.34375

A C     -0.85156   0.0   0.75   0.0    17
A C     -0.85156   0.0   0.75   343    360.0

N        0.0       0.0   B411   ROMAN  3.0    0.0
```

Example 2: BS321 - Park Exit



File: BS321KC.SDF

```
L P C      2.0   2.0   0.0          0.0          0.0          0.0625
L P C      2.0   2.0   0.0          0.125        0.0          0.25
L P C      2.0   2.0   0.0          0.3125       0.0          0.4375
L P C      2.0   2.0   0.0          0.5          0.0          0.625
L P C      2.0   2.0   0.0          0.6875       0.0          0.8125
L P C      2.0   2.0   0.0          0.875        0.0          0.9375
L P C      2.0   2.0   0.0          0.0          0.0          -0.0625
L P C      2.0   2.0   0.0          - 0.125      0.0          -0.25
L P C      2.0   2.0   0.0          -0.3125      0.0          -0.4375
L P C      2.0   2.0   0.0          -0.5         0.0          -0.625
L P C      2.0   2.0   0.0          -0.6875      0.0          -0.8125
L P C      2.0   2.0   0.0          -0.875       0.0          -0.9375
L P C      1.0   1.0   -0.03125     0.0          -0.03125     0.0625
L P C      1.0   1.0   -0.09375     0.0          -0.03125     0.0625
L P C      1.0   1.0   -0.03125     0.125        -0.03125     0.25
L P C      1.0   1.0   -0.09375     0.1875       -0.03125     0.125
L P C      1.0   1.0   -0.09375     0.1875       -0.03125     0.25
L P C      1.0   1.0   -0.03125     0.3125       -0.03125     0.4375
L P C      1.0   1.0   -0.09375     0.375        -0.03125     0.4375
L P C      1.0   1.0   -0.09375     0.375        -0.03125     0.3125
L P C      1.0   1.0   -0.03125     0.5          -0.03125     0.5625
L P C      1.0   1.0   -0.09375     0.5625       -0.03125     0.5
L P C      1.0   1.0   -0.03125     0.0          -0.03125     -0.0625
L P C      1.0   1.0   -0.09375     0.0          -0.03125     -0.0625
L P C      1.0   1.0   -0.03125     -0.125       -0.03125     -0.25
```

```
L P C       1.0   1.0   -0.09375    -0.1875     -0.03125    -0.125
L P C       1.0   1.0   -0.09375    -0.1875     -0.03125    -0.25
L P C       1.0   1.0   -0.03125    -0.3125     -0.03125    -0.4375
L P C       1.0   1.0   -0.09375    -0.375      -0.03125    -0.4375
L P C       1.0   1.0   -0.09375    -0.375      -0.03125    -0.3125
L P C       1.0   1.0   -0.03125    -0.5        -0.03125    -0.625
L P C       1.0   1.0   -0.09375    -0.5625     -0.03125    -0.625
L P C       1.0   1.0   -0.09375    -0.5625     -0.03125    -0.5

V C   90.0  -0.046875   -0.9375   -0.046875   -0.9375 STANDARD    0.0625    0.0
```

7.0 APPENDIX B

GLOSSARY OF VARIABLE NAMES

7.1 ARI DIAGRAM PROCESSING

A

a_notation: Text string for AT and BT features
a_symbolno: Symbol number of AT and BT features
area[]: Array of structure to store area information
areainfo[]: Array of structure to store area information
areanames[]: Array of characters for actual areas
areaname[]: Array of characters for actual areas
areas[]: Array of structure to store area information

B

begdraw: Beginning of an actual drawing area in a page

C

cltrfile: File pointer of clutter symbol file
complex_name: Complex interchange symbol name
composite_symbols[]: Array of composite text symbols
count: Integer

D

datafile: File name
dxffile: Header file for a DXF file

E

entfile: Entity file for a DXF file

F

feature: Structure to store feature and its area information
field: Array of characters which returns the composition of a text symbol
fp: File pointer

H

Height: Area height
hoff: Horizontal offset of a diagram

I

input_rec: Structure to store data read from datafile
level: MicroStation layer

M

mileincr: Mile increment

N

new_sheet: Integer
new_xmax: Xmax for graphic feature
new_xmin: Xmin for graphic feature
new_ymax: Ymax for graphic feature
new_ymin: Ymin for graphic feature
no_areas: Integer
no_of_simple_textsym: Integer
no_symbols: Integer

O

outfile: File pointer

P

prev_enddraw: End of previous diagram (page)
prev_xbegdraw: Beginning of previous diagram (page)

R

rdbd: Roadbed
rdbd_id: Roadbed
rec_info: Structure to store data read from datafile

S

sdffile: Symbol description format filename
sheet: Integer
sheet_no: Integer
sheetno: Integer
simple_symbols[]: Array of normal/cryptic symbol numbers
skew: Angle of intersection

symbol_name: Feature name
symbolname: Feature name
symbolno: Feature name
symfile: Symbol description format filename

T

txtstr: Text string present in a feature

V

V_offset: Vertical offset of an area
voff: Vertical offset of a diagram

X

xbegform: Beginning X coordinate of a form
xbegmile: Beginning mile on a page
xendmile: Ending mile on a page
xoff: Horizontal offset of a diagram

Y

yoff: Vertical offset of a diagram
yoffset: Center of the vertical axis

7.2 CLUTTER DETECTION AND PROCESSING

A

area[]: Array of structure to store area information
area_loc: Area where 'N' text is placed
areainfo[]: Array of structure to store area information
areanames[]: Array of characters to store actual area
areas[]: Array of structure to store area information

B

boundry_set[]: Array of boundary symbols

C

composite_symbols[]: Array of composite text symbols
count: Integer

D

datafile: File name

F

fclutter: File pointer of clutter symbol file
feature: Structure to store feature name and its area info
field: Array of characters which returns the composition of a text symbol
fp: File pointer

I

input_rec: Structure to store data read from datafile

N

new_xmax: Xmax for graphic feature
new_xmin: Xmin for graphic feature
new_ymax: Ymax for graphic feature
new_ymin: Ymin for graphic feature
no_areas: Integer
no_of_notations: Integer
no_of_set: Integer
no_of_simple_textsym: Integer
no_symbols: Integer

P

prev_graphic_rec: Previous graphic feature record
prev_input_rec: Previous record
prev_text_rec: Previous text record

R

rec_info: Structure to store data read from datafile

S

sdffile: Symbol description format file name
setno: Integer
sheet: Integer
sheet_no: Integer
simple_symbols[]: Array of normal/cryptic symbols names
symbol_name: Feature name
symbolno: Feature name
symfile: Symbol description format file name

T

txtstr: Text string present in a feature

X

xoff: Horizontal offset of a diagram

Y

yoff: Vertical offset of a diagram