FINAL REPORT

COST AND ENERGY EFFICIENT SUBDIVISION DESIGN: THE RESIDENTIAL AREA COST ANALYSIS MODEL (RACAM)

PREPARED BY:

F.A. CURTIS
PRINCIPAL INVESTIGATOR
FACULTY OF ENGINEERING
UNIVERSITY OF REGINA
REGINA, SASKATCHEWAN
S4S 0A2

FOR:

EXTERNAL RESEARCH PROGRAM
CANADA MORTGAGE AND HOUSING CORPORATION
OTTAWA, ONTARIO
CANADA

AUGUST, 1987

RACAM

(Residential Area Cost Analysis Model)

REFERENCE MANUAL

VERSION 6.0

FACULTY OF ENGINEERING

UNIVERSITY OF REGINA

REGINA, SASKATCHEWAN, CANADA

AUGUST, 1987

Principal Investigator: Professor Fred Curtis

Research Assistants: Michael J. Lyzaniwski
Barry M. Rezansoff
Michael W. Schlosser

"This project was carried out with the assistance of a grant from the Canadian Mortgage and Housing Corporation under the terms of the External Research Program. The views expressed are those of the author and do not represent the official views of the Corporation."

Acknowledgements

The principal investigator would like to thank Michael J. Lyzaniwski and Barry Rezansoff, as well as Michael W. Schlosser for their contributions and extensive research undertaken in the preparation of this document.

RBASE/RACAM INPUT METHOD MANUAL

Table of Contents Page

Introduction

This manual must initially be used in conjunction with the racam manual in order to explain the calculation and variable parameters of Racam. They have not been repeated in this manual.

The purpose of the input routine or objective is to allow a more efficient input operation. One that is less error prone and more easily subject to analyst manipulation. A second interim objective is to make the RACAM package partially PC (Personal Computer) based. While the number crunching and the calculations are currently done on a VAX, the data manipulation and preparation is done on the PC.

Rbase Explanation

Rbase is a PC based database system with its own procedural language. The RACAM application built from it is a set of menus, information screens and commands contained in a Rbase program. The advantage of the system is that the user needs little training to intiate work on the system. Many of the functions and commands are self explanatory.

This Rbase application is a data input and management system.

The menus are choices the user is allowed to make in order to enter new data or alter existing data.

The information screens and commands are the data entry screens that are displayed when the user has completed his menu choices, or the brief help screens summarizing the functions of the screen.

Screens & Functions

The following is a Schematic Flowchart of menu choices. (N.B. All screens are equipped with a help screen by pressing the F10 key on the keyboard.)

```
Racam Main Menu
| <----------> Subdivision Data Entry and Modification
|                    |   <-----------> Account Percentages
|                    |
|                    |   <-----------> Construction Costs
|                    |                 <-----> Interest Rate &
|                    |                         Subdivision Life
|                    |                 <-----> Maintenance Costs
|                    |                         by Subdivision Life
|                    |                 <-----> Operational Costs
|                    |                         by Subdivision Life
|                    |                 <-----> Exit
|                    |
|                    |   <-----------> Depth and Diameter Data
|                    |                 <-----> Watermain Unit
|                    |                         Costs
|                    |                 <-----> Domestic Sewer Unit
|                    |                         Costs
|                    |                 <-----> Storm Sewer Unit
|                    |                         Costs
|                    |                 <-----> Catchbasin Lead
|                    |                         Unit Costs
|                    |                 <-----> Exit
|                    |
|                    |   <-----------> Subdivisions Areas
|                    |
|                    |   <-----------> Unit Costs
|                    |                 <-----> Street Requirement
|                    |                         Unit Costs
|                    |                 <-----> Manhole and
|                    |                         Catchbasin Unit
|                    |                         Costs
|                    |                 <-----> Dwelling and
|                    |                         Maintenance
|                    |                         Unit Costs
|                    |                 <-----> Racam Calibration
|                    |                         Values
|                    |                 <-----> Exit
|                    |
|                    | <-----------> Link Data
|                    |                 <-----> Link Data 1
|                    |                 <-----> Link Data 2
|                    |                 <-----> Link Data 3
|                    |                 <-----> Link Data 4
```

```
|                    |                <-----> Exit
|                    |
|                    | <----------> Node Data
|                    |
|                    | <----------> Exit
|                    |
| <---------> Reports
|                    | <----------> Output
|                    |
|                    | <----------> Exit
|                    |
| <---------> Exit
```

Subdivision Data Entry and Modification

This is the first menu screen that user sees upon execution of the Rbase program. It initially gives the user the choice to either:

1. Subdivision Data Entry and Modification:
   - enter and/or alter data,

2. Reports:
   - produce the required file for uploading,

3. Exit:
   - leave the program.

Choosing option 1. Subdivision Data Entry and Modification, the user has 7 data entry options available to him in order to facilitate efficient and effective data management.

1. Account Percentages
2. Construction Costs
3. Depth and Diameter Data
4. Subdivisions Areas
5. Unit Costs
6. Link Data
7. Node Data

1. Account Percentages

When the user makes this choice he is immediately shown the screen which edits the Account percentages:

```
Subdivision..............
Element Number...........
Element Name.............
Account Number...........
Account Name.............
Percentage...............
```

After this screen the Escape key causes the accperc table to be shown in order to directly alter the data in the database. Hitting the escape key again will return the user to the Subdivision Data Entry and Modification screen.

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Element number and element name identify the subdivision elements: (ie. street lights, pavement etc.)

Account number and account name identify the account that will be charged a particular percentage of the construction and or maintenance and operating costs.

Percentage is that percentage of the total costs that the account will be charged of the entire elements cost.

There are three options (in addition to the exit option) available under the construction cost heading:

```
Construction Costs
   <-----> Interest Rate &
           Subdivision Life
   <-----> Maintenance Costs
           by Subdivision Life
   <-----> Operational Costs
           by Subdivision Life
```

They all display the following screen which facilitates altering the concost table that stores the interest rate, maintenance and operational cost data:

```
Subdivision............
Status.................
Element Number.........
Element Name...........
Cost...................
Gradient Factor........
Interest Rate..........
Life of Element........
```

After this screen the user has the option upon hitting the escape key which will display the concost table and allows direct alteration of it.

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Status is the variable that identifies whether the data entered is for the Interest Rate & Subdivision Life, or the Maintenance Costs by Subdivision Life, or Operational Costs by Subdivision Life.

Element number and element name identify the subdivision elements: (ie. street lights, pavement etc.)

Cost is the price of the element, either construction, maintenance, or operational cost.

Gradient Factor is the financial accounting variable that allows calculation of the annual costs of maintaining or paying for the element over its lifetime.

Interest rate is the rate of interest the element or subdivision will be charged over the life of the element or subdivision.

Life of the element is the expected seviceable life of the element.

Choosing the third option allows the user to alter or enter depth and diameter data. Four choices are available (along with the exit option) for the user to alter the dpthdia table which contains cost tables for various lengths, diameters and permissible operating depths of pipes.

Depth and Diameter Data

```
<-----> Watermain Unit Costs
<-----> Domestic Sewer Unit Costs
<-----> Storm Sewer Unit Costs
<-----> Catchbasin Lead Unit Costs
```

Choosing any of the above options will display the following screen:

```
Subdivision Name........
Element Number..........
Element Name............
Diameter Index......(or Depth Index for the Catchbasin option)
Diameter Range.....[1](for the Watermain option)
Unit Cost...............
```

1. Depth Index for the Domestic sewer option
   Depth Number for the Storm sewer option
   Depth Range for the Catchbasin option

After this screen the user has the option upon hitting the escape key which will display the dpthdia table and allows direct alteration of it.

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Element number and element name identify the subdivision elements: (ie. street lights, pavement etc.)

Depth or Diameter index, range, number is the category by which the watermain, storm or domestic sewer and catchbasin lead pipes are separated for costing purposes. They indicate a range of pipe depths or diameters that together with another depth or diameter dimension will cost the pipe. The record contains the smallest dimension in the range. The cost is relevant up to the next range.

Unit cost is the per meter cost of the pipe for the two dimension given in the previous variable.

Subdivisions Areas

Choosing the Subdivision Areas option displays the following screen for alteration of the subareas table. This table contains the elements that are not covered in the other tables.

Subdivisions Areas

Subdivision.....
Element Number.....
Element Name.....
Area.....

After this screen the user has the option upon hitting the escape key which will display the subareas table and allows direct alteration of it.

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Element number and element name identify the subdivision elements: (ie. street lights, pavement etc.)

Area is the dimension (usually hectares) that the element covers (ie. Park, Schools, etc.).

Unit Costs

Choosing this option allows the user to alter the unit costs of the following four elements (along with the exit option):

Unit Costs
1. Street Requirement Unit Costs
2. Manhole and Catchbasin Unit Costs
3. Dwelling and Maintenance Unit Costs
4. Racam Calibration Values

All four options display the following screen for the alteration of the unitcost table:

Subdivision Name.....
Element number.....
Element name.....
Requirement Number.....
Unit cost.....

After this screen the user has the option upon hitting the escape key which will display the dpthdia table and allows direct alteration of it.

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Element number and element name identify the subdivision elements: (ie. street lights, pavement etc.)

Requirement number is an unused variable at the current time. In the future it could be used to group together different classes

of unit costs for use in analyzing different link/node combinations.

Unit cost is the dollar cost of the element.

Link Data

This option allows alteration of the link data for the subdivision in the linkdata table. Four screens are available (along with the exit option).

After these screens the user has the option upon hitting the escape key which will display the linkdata table and allows direct alteration of it.

Link Data
1. Link Data 1
2. Link Data 2
3. Link Data 3
4. Link Data 4

The Link Data 1 screen displays the following:

Subdivision.....
Origin.....
Destination .....
Length of link.....
Number of Singles.....
Number of Multis.....
Area.....
Saleable Frontage.....

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Origin identifies the origin node of the particular link in question.

Destination identifies the destination node of the particular link in question.

Length of link is in meters.

Number of singles/multis quantifies the number of single and multi-family dwellings on the link.

Area is the physical area accessed by the link.

Saleable frontage based on the design of the link and is in meters.

The Link Data 2 screen displays the following:

Subdivision Name.....
Road length.....
Road Width.....
Road Type.....

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Road length and width are in meters and measured form node to node for length and as a standard or average for width.

Road type is Racam code to identify various types of roads. .(ie. those leading to a cul-de-sac, etc.)

The Link Data 3 screen displays the following:

Subdivision Name.....
Sidewalk Length.....
Barrier Length.....
Rolled Length.....
Boulevard Length.....

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

All dimensions are in meters. The barrier and rolled lengths refer to the curb types.

The Link Data 4 screen displays the following:

Subdivision Name.....
Link number.....
Watermain Population.....
Domestic Sewer Population..
Storm Sewer Area (ha.).....

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Link Number is an unused variable at the present time.

Watermain population indicates the estimated population of the link using water. This number is used for watermain calculations. The same for the domestic sewer population.

Storm sewer area is used to estimate runoff water amounts for storm sewer calculations.

Node Data

This option allows alteration of the node data for the subdivision in the nodedata table. Choosing this option will display the following screen:

Node Data

Subdivision Name.....
Node Type.....
Node Elevation.....
Radius.....

After these screens the user has the option upon hitting the escape key which will display the linkdata table and allows direct alteration of it.

Subdivision or Subdivision Name is the key variable that links all of the relevant subdivision data together. All data rows have this common value in order to identify the data set that deals with a particular subdivision.

Node type is the Racam code differentiating various nodes.

Node Elevation is the physical elevation of the node, again in meters and used for pipe costing.

Radius is the curve radius of the pavement at the corner or in the cul-de-sac. If the node is not at a curve the radius is zero.

Output

There is only one option in the Output screen that which produces the PC file that will be uploaded to the VAX for processing.

4. Reports

The report section of the Rbase program automatically produces a file that can be uploaded for the execution of the Racam Program on the University of Regina's VAX.

5. Upload Procedure

The procedure to upload is as follows:

This is assuming the kermit package of software resides on the users current path/directory and the coax or other cable is plugged into the PC.

1. Execute the Kermit command and logon to the VAX machine and the users meena account.

2. Execute the kermit command in the host and execute the receive command in order to be in the recieve mode.

3. Press Control C (^C) to escape to the PC session and execute the send (filename) command.

4. The PC will display a screen that will indicate the status of the operation and will also indicate completion.

TABLE OF CONTENTS

CHAPTER 3 RACAM OPERATION

CHAPTER 4 RACAM OUTPUT

CHAPTER 5 DESCRIPTION OF RACAM MODULES

APPENDIX A SAMPLE DATA ACQUISITION SHEETS

APPENDIX B PROGRAM LISTING

APPENDIX C SAMPLE DATA FILES

APPENDIX D SAMPLE OUTPUT FILE

CHAPTER 1

INTRODUCTION

1.1 WHAT IS RACAM?

The determination of the most cost efficient subdivision development requires a comparison of alternative subdivision layouts. This activity involves an assessment of construction, operation and maintenance costs for subdivision service elements, including water, sewer, utilities, roads and public open spaces.

The Residential Area Cost Analysis Model (RACAM) was developed, and computerized to assess the construction, operation and maintenance costs for a proposed subdivision plan. RACAM calculates present and annual costs.

RACAM is part of a larger computer procedure used for subdivision development. This urban modelling and design procedure, being developed at the University of Regina, includes several models in the pre-design, design, evaluation and management phases of subdivision development. (Figure 1)

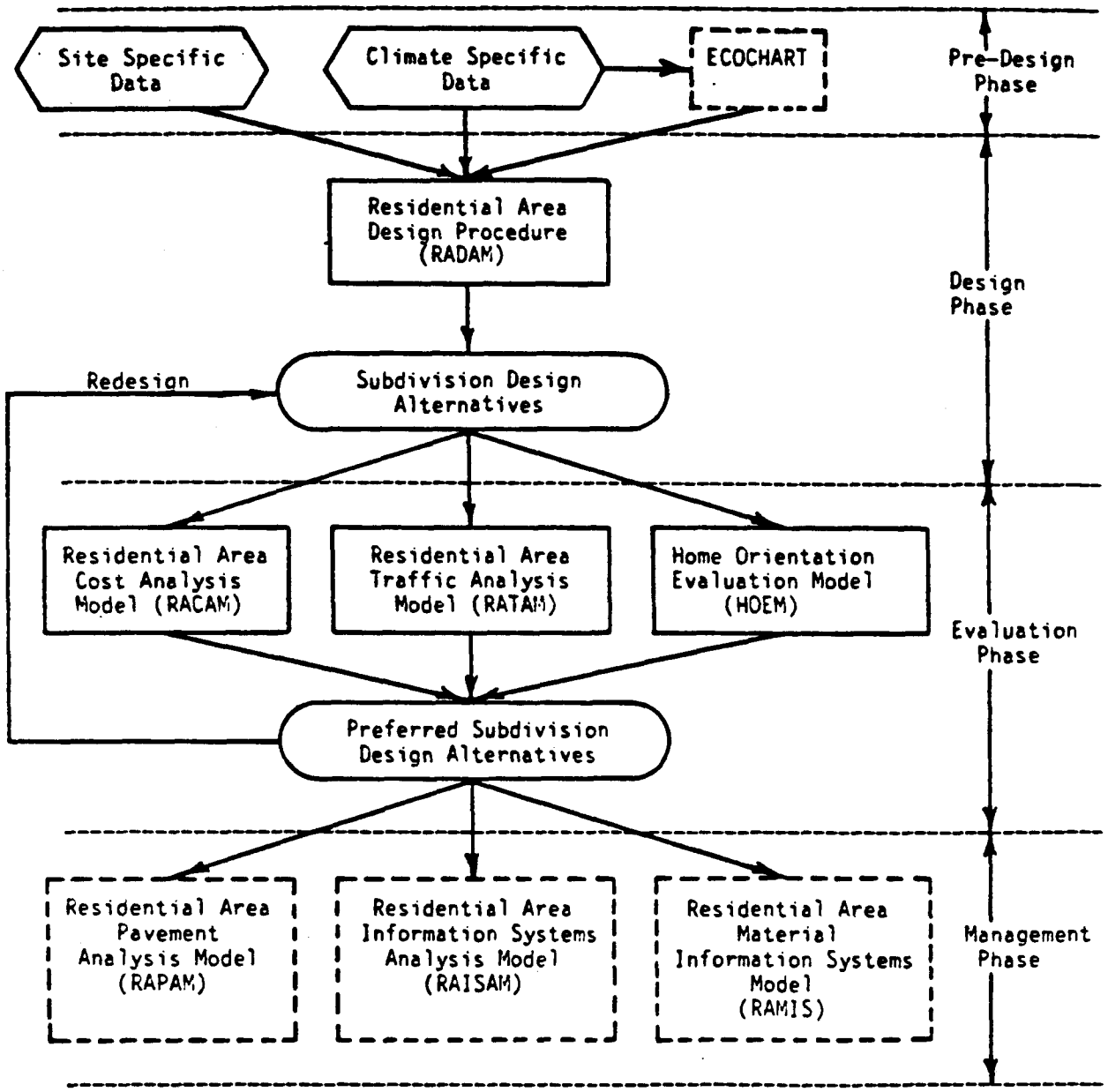


Figure 1, Urban Design and Modelling Procedure

1.2 HARDWARE REQUIREMENTS

RACAM is written to operate on a VAX 8600 running the VMS operating system. If changes to the original RACAM code are required, a VAX FORTRAN compiler must be accessible on the VAX. RACAM should be executed using a VT-100 type terminal.

CHAPTER 2

RACAM DATA REQUIREMENTS

2.1 OVERVIEW

RACAM requires that a subdivision layout be converted to a link and node format. Nodes represent points on the subdivision plan (such as corners,intersections and exits), and links represent lines connecting these nodes (including thoroughfares and easements). The link and node data required by RACAM, including link and node characteristics and how the links and nodes are connected, are provided in a data file for each subdivision layout.

Economic data is also required by RACAM. This information, including unit cost information for subdivision service elements, is also provided in a data file, distinct from the link and node data. The economic data file is specific to the cost data in the municipality where the subdivision is being planned.

2.2 RACAM INPUT DATA FILES

RACAM requires that the two data files, the economic and link/node data files, be created before the program is executed. Use the VAX/VMS editor facility to create these data files.

2.2.1 ECONOMIC DATA FILES:

CREATING A NEW ECONOMIC DATA FILE

Before creating a new economic data file, it is necessary to logon (or gain access) to the University of Regina's VAX 8600 computer, Meena, from a VT-100 terminal.

LOGIN PROCEDURE

Press the <RETURN> key on the terminal twice. After a beep, the following prompt will appear:

REQUEST:

When this prompt appears on the screen, type the following:

MEENA <RETURN>

After a beep the following prompt will appear:

USERNAME:

The user should enter the name of the account which contains the RACAM program. Presently, it resides in the CURTIS account and can be accessed by entering:

CURTIS <RETURN>

The password prompt will then appear:

PASSWORD:

This password will be given individually to authorized users. Type the password and press the <RETURN> key to obtain access and to logon to the account. Some system messages will appear on the screen, followed by the dollar sign prompt ("$"). The login procedure is now complete.

ENTERING ECONOMIC DATA

To begin entering economic data, enter the following at the "$" prompt:

EDIT filename.DAT

Use a filename that can be easily remembered and relates to the area under consideration (eg. Use ECONREGNA.DAT as a filename for the economic data file for Regina.)

After entering this command, the following will appear on the screen:

[EOB]

At the bottom of the screen will be the message:

Input file does not exist

The following economic data can now be entered. (See Section C.1, "SAMPLE DATA FILES: ECONOMIC DATA".)

i) INTEREST RATE AND SUBDIVISION LIFE

The interest rate and corresponding subdivision life data indicates the expected service lifespan of a subdivision for different interest rates, and is required for present worth and annual cost calculations. The information in this section includes the following data:

INTEREST RATE - the interest rate used for present value calculation of construction costs. Current or projected bank interest rates can be used for the interest values.

SUBDIVISION LIFE - the projected lifespan of the entire subdivision. For each of the interest rates described above, enter a corresponding subdivision life.

Compile the data for this section on the "INTEREST RATE AND SUBDIVISION LIFE" data sheet (Section A.1.1). Enter the data into the economic data file from the data sheet according to the following format:

INTEREST RATE (F10.2) SUBDIVISION LIFE (I10)

The format code "F10.2" indicates that the data entered in the "INTEREST RATE" column is ten digits long (including the decimal) and two decimals places will be assumed if no decimal point is specified (eg 5025 will be read as 50.25). The format code "I10" indicates that the values entered in

the "SUBDIVISION LIFE" column are integer values and are ten digits long.

When all interest rate and subdivision life values have been entered, enter the flag values "-1." and "-1" in the "INTEREST RATE" and "SUBDIVISION LIFE" columns respectively (Note the decimal point in the first flag value). This section of the economic data file, when complete, should look like:

```
*******5.0********50
*******7.0********75
*******8.0*******100
        .
        .
        .
*******-1.********-1  } flag values
```

with the the "*" characters indicating spaces.

ii) STREET REQUIREMENTS

This information describes unit costs for the various street requirements, such as traffic signs and sidewalks. The following is a description of each of the unit costs in this section:

a) STREET SIGN ($/SIGN) - material and construction cost for installing one street sign

b) STREET LIGHT ($/LIGHT) - material and construction cost for one street light

c) PARK DEVELOPMENT ($/HECTARE) - unit construction cost for park land. This may include the laying of sod or planting of grass, planting of trees, the placement of sand or crusher dust, or the installation of playground facilities.

d) BUFFER DEVELOPMENT ($/HECTARE) - unit material and construction costs for buffer land. This may include the laying of sod or planting of grass, or the planting of trees.

e) COLLECTOR STREET ($/$m^2$) - base and pavement material and construction costs for a unit of collector street area

f) LOCAL STREET ($/$m^2$) - base and pavement material and construction costs for a unit of local street area. This cost will be somewhat less than that for collector

streets above due to thinner base and pavements requirements.

g) BARRIER CURB AND GUTTER ($/m) - material and construction costs for a linear meter of barrier curb and gutter (Figure 2, Barrier Curb and Gutter).

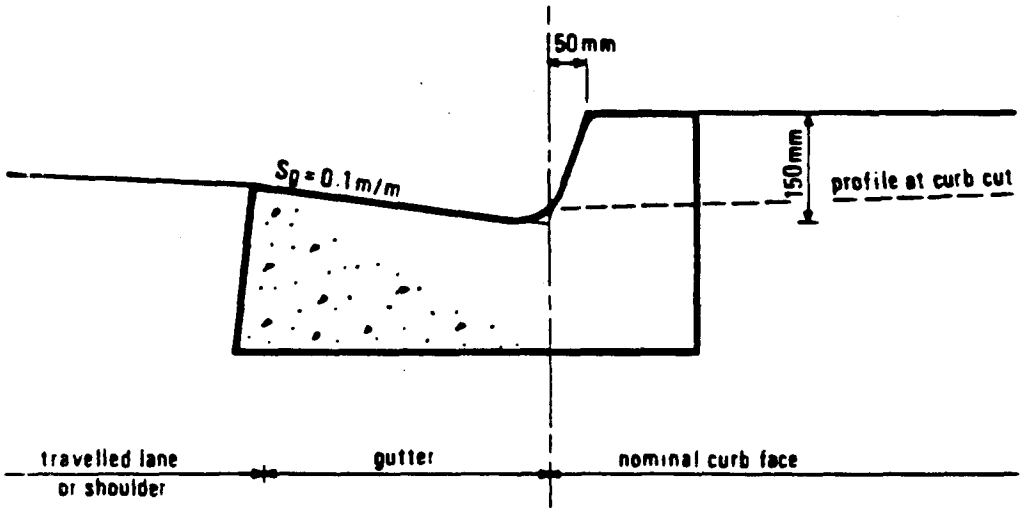


Figure 2, Barrier Curb and Gutter

h) ROLLED CURB AND GUTTER ($/m) - material and construction costs for a linear meter of rolled curb and gutter (Figure 3, Rolled Curb and Gutter).

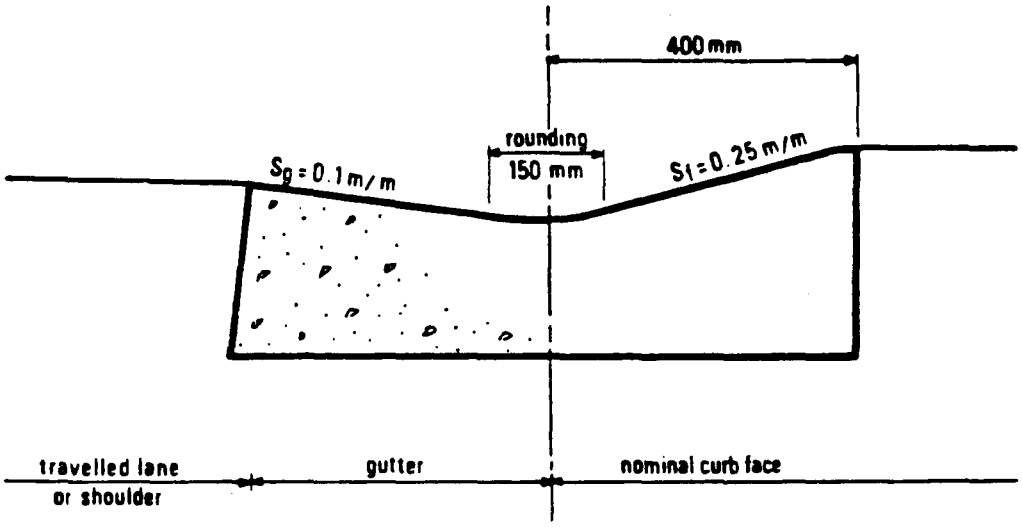
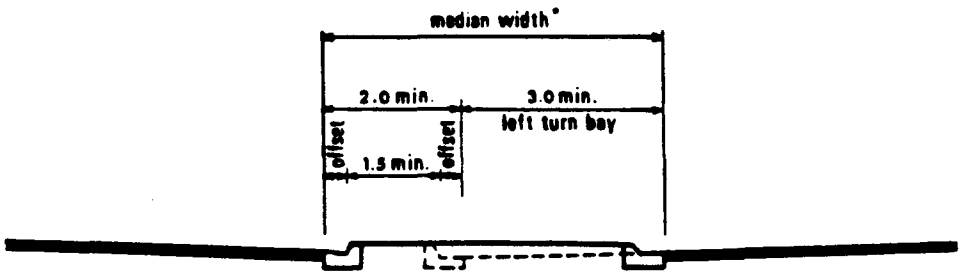


Figure 3, Rolled Curb and Gutter

i) BOULEVARD CURB AND GUTTER ($/m) - material and construction costs for a linear meter of boulevard curb and gutter (Figure 4, Boulevard Curb and Gutter).



* median width 5.0 m with 0.25 m offset and provision for 3.0 m left turn bay.
median width 5.5 m with 0.50 m offset and provision for 3.0 m left turn bay.
cross slope 0.01 m/m minimum, 0.03 m/m maximum.

Figure 4, Boulevard Curb and Gutter

Compile this data on the "STREET REQUIREMENT UNIT COST" data sheet (Section A.1.2). Enter the data into the data file in the same order as on the data sheet and according to the following format:

```
$/STREET SIGN  (F8.2)
$/STREET LIGHT (F8.2)
        .
        .
        .
```

The format code "F8.2" indicates that the data entered for this section is eight digits long (including the decimal point), and that two decimal places will be assumed if no decimal point is entered. When complete, this section of the economic data file should look like:

```
***10.00
***20.00
    .
    .
    .
```

where the "*" characters indicate spaces.

iii) WATERMAIN UNIT COSTS

This data describes the unit costs of watermain pipe for six different pipe diameter ranges. Each pipe diameter range corresponds to a watermain diameter index:

DIAMETER INDEX	DIAMETER RANGE (mm)
1	0 - 150
2	151 - 200
3	201 - 250
4	251 - 300
5	301 - 400
6	401 - 9999

Compile the data for this section on the "WATERMAIN UNIT COST" data sheet (Section A.1.3). Enter the data in the economic data file in the same order as on the data sheet and according to the following format:

COST1 (F8.2) COST2 (F8.2) COST3 (F8.2)...COST6 (F8.2)

the format code "F8.2" being described above. When complete, this section of the data file should look like:

10.0014.00***15.00***12.00***11.00***13.00

where the "*" characters indicate spaces.

iv) DOMESTIC SEWER UNIT COSTS

This data describes the unit costs for domestic sewer pipes for five different diameter ranges and nine different depth ranges. Each diameter range has a corresponding domestic sewer diameter index:

DIAMETER INDEX	DIAMETER RANGE (mm)
1	0 - 200
2	201 - 250
3	251 - 300
4	301 - 375
5	GREATER THAN 375

As well, each depth range has a corresponding domestic sewer depth index:

DEPTH INDEX	DEPTH RANGE (m)
1	0 - 3.0
2	3.1 - 3.5
3	3.6 - 4.0
4	4.1 - 4.5
5	4.6 - 5.0
6	5.1 - 5.5
7	5.6 - 6.0
8	6.1 - 6.5
9	Out of Bounds

Compile the data for this section on the "DOMESTIC SEWER UNIT COST" data sheet (Section A.1.4). Enter the data in the economic data file in the same order as on the data sheet and according to the following format:

```
cost(depth1,diameter1)(F8.2)...cost(depth9,diameter1)(F8.2)
          .                                   .
          .                                   .
          .                                   .
cost(depth1,diameter5)(F8.2)...cost(depth9,diameter1)(F8.2),
```

5

the format code "F8.2" described above. When complete, this section of the data file should look like:

```
***10.00***12.00***13.00****9.00***13.00...14.00
***12.00***14.00***11.00***11.00***19.00...17.00
     .       .       .       .       .       .
     .       .       .       .       .       .
     .       .       .       .       .       .
***11.00***12.00***14.00***13.00***15.00...11.00
```

where the "*" characters indicate spaces.

v) STORM SEWER UNIT COSTS

This data describes the unit costs of storm sewer pipes for fourteen different diameter ranges and nine different depth ranges. Each diameter range has a corresponding storm sewer diameter index:

DIAMETER INDEX	DIAMETER RANGE (mm)
1	0 - 300
2	301 - 375
3	376 - 450
4	451 - 525
5	526 - 600
6	601 - 675
7	676 - 750
8	751 - 900
9	901 - 1050
10	1051 - 1200
11	1201 - 1350
12	1351 - 1500
13	1501 - 1650
14	1651 - 1800

Also, for each depth range, there is a corresponding storm sewer depth index (See Section iv, "DOMESTIC SEWER UNIT COSTS" for the depth indexes corresponding to the nine depth ranges).

Compile this data on the "STORM SEWER UNIT COSTS" data sheet (Section A.1.5). Enter the data into the economic data file in the same order as on the data sheet, and according to the following format:

```
cost(depth1,diameter1)(F8.2)...cost(depth9,diameter1)(F8.2)
          .                                   .
          .                                   .
          .                                   .
cost(depth1,diameter14)(F8.2)..cost(depth9,diameter14)(F8.2)
```

the format code "F8.2" being described above. When complete, this section of the data file should look like:

```
***10.00***12.00***13.00****9.00***13.00...14.00
***12.00***14.00***11.00***11.00***19.00...17.00
     .       .       .       .       .       .
     .       .       .       .       .       .
     .       .       .       .       .       .
***11.00***12.00***14.00***13.00***15.00...11.00
```

where the "*" characters indicate spaces.

vi) MANHOLE AND CATCHBASIN UNIT COSTS

This data describes the costs for catchbasin and manhole components. This section of the economic data file includes the following unit costs:

a) MANHOLE: BASE, COVER, RISER, FRAME ($/MANHOLE) - cost for the above-ground portion of the manhole. This does not include the manhole shaft.

b) MANHOLE SHAFT ($/m-DEPTH) - cost per meter for the shaft leading down from the manhole.

c) CATCHBASIN: FRAME, COVER ($/CATCHBASIN) - cost for the above-ground portion of the catchbasin. This does not include the catchbasin shaft.

d) CATCHBASIN SHAFT ($/m-DEPTH) - cost per meter for the shaft leading down from the catchbasin.

Compile the data for this section on the "MANHOLE AND CATCHBASIN UNIT COST" data sheet (Section A.1.6). Enter the data in the economic data file in the same order as on the data sheet, and according to the following format:

```
COST1 (F8.2)
   .
   .
   .
COST4 (F8.2),
```

the format code "F8.2" being described above. When complete, this section of data should look like:

```
**900.00
**150.00
**800.00
**900.00
```

where the "*" characters indicate spaces.

vii) CATCHBASIN LEAD UNIT COSTS

This data describes the unit costs of catchbasin leads, which are pipes leading from the catchbasins to the storm sewer pipes. Unit costs for catchbasin leads are required for nine different depth ranges. For each depth range, there is a corresponding depth index (See Section iv, "DOMESTIC SEWER UNIT COSTS" for the depth indexes corresponding to the nine depth ranges).

Compile the data for this section on the "CATCHBASIN LEAD UNIT COSTS" data sheet (Section A.1.7). Enter the data into the economic data file in the same order as on the data sheet, and according to the following format:

COST1 (F8.2) COST2 (F8.2)...COST9 (F8.2),

the format code "F8.2" being described above. When complete, this section of the data file should look like:

10.0011.00**15.00...14.00

where the "*" characters indicate spaces.

viii) **DWELLING AND MAINTENANCE UNIT COSTS:**

This data describes the unit costs for installation of services (gas, power, phone) per dwelling, as well as unit maintenance costs for the subdivision. This section of the economic data file includes the following:

a) POWER: SINGLE-FAMILY UNIT ($/UNIT) - the construction and material costs for providing power to a single-family dwelling unit

b) POWER: MULTI-FAMILY UNIT ($/UNIT) - the construction and material costs for providing power to a multi-family dwelling unit

c) GAS: SINGLE-FAMILY UNIT ($/UNIT) - the construction and material costs for providing gas for heating to a single-family unit

d) GAS: MULTI-FAMILY UNIT ($/UNIT) - the construction and material costs for providing gas for heating to a multi-family unit

e) PHONE: SINGLE-FAMILY UNIT ($/UNIT) - the construction and material cost for providing telephone facilities to a single-family unit

f) PHONE: MULTI-FAMILY UNIT ($/UNIT) - the construction and material cost for providing telephone facilities to a multi-family unit

g) STREET SWEEPING ($/Km/YR) - the yearly street-sweeping cost per kilometer of subdivision street

h) WINTER ROAD MAINTENANCE ($/Km/YR) - the yearly cost for ploughing, sanding and salting a kilometer of subdivision street

i) ASPHALT MAINTENANCE ($/Km/YR) - includes the filling of potholes and repairing of cracks in subdivision streets

j) CONCRETE MAINTENANCE ($/Km/YR) - includes the filling of potholes, regrading of worn surfaces and repairing of cracks for sections of concrete (sidewalks,curbs) in the subdivision

k) SEWER MAINTENANCE ($/Km/YR) - includes the removal of debris, annual cleaning of pipes, opening of fire hydrants to clear lines, and the checking sewer valves

l) SOLID WASTE COLLECTION ($/HECTARE/YR) - annual garbage collection cost

m) STREET LIGHT MAINTENANCE ($/LIGHT/YR) - annual cost for repair or replacement of damaged street lights

n) PARK AND BUFFER MAINTENANCE ($/HECTARE/YR) - includes the cost of such things as mowing and watering lawns

Compile this data on the "DWELLING AND MAINTENANCE UNIT COSTS" data sheet (Section A.1.8). Enter the data into the economic data file in the same order as on the data sheet and according to the following format:

```
COST1 (F8.2)
COST2 (F8.2)
   .
   .
   .
COST15 (F8.2)
```

the format code "F8.2" being described above. When complete, this section of the data file should look like:

```
**112.00
**150.00
***99.00
      .
      .
      .
**190.00
```

where the "*" characters indicate spaces.

ix) RACAM CALIBRATION VALUES

This data describes all the calibration values used in RACAM. These calibration values may be obtained from municipal engineering departments, or the RACAM defaults may be used as described below. The data input in this section of the economic data file includes:

a) POPULATION DENSITY (PEOPLE/UNIT) - the average number of people living in each dwelling unit. If 0.0 is entered for this value, the default value of 3.35 people/unit is assumed.

b) WATERMAIN MINIMUM VELOCITY (m/s) - the minimum allowable water velocity in any watermain pipe. This value is determined by local standards and geography. If 0.0 is entered for this value, the default value of 1.75 m/s is assumed.

c) MANNING'S ROUGHNESS (unitless) - indicates the roughness of the material being used for sewer and watermain pipes. This value can be found for different materials in most fluid mechanics texts or from local suppliers. If 0.0 is entered for this value, the default value of 0.013 is assumed.

d) DOMESTIC WATER USE (m^3/PERSON/DAY) - the estimated daily water consumption for each person in the subdivision. If 0.0 is entered for this value, the default value of 0.5678 m^3/person/day is assumed.

e) WATER USE SAFETY FACTOR (unitless) - used in the design of watermain and domestic sewer systems to allow for peak periods of water use. The value specified here is the number of times the domestic water use (above) the watermain and domestic sewer systems should be designed for. If 0.0 is entered for this value, the default value of 2 is assumed.

f) DOMESTIC SEWER COVER DEPTH (m) - the minimum amount of ground cover for domestic sewer pipes. If 0.0 is entered for this value, the default value of 2.75 m is assumed.

g) DOMESTIC SEWER MINIMUM VELOCITY (m/s) - the minimum allowable velocity in any domestic sewer pipe. This value is determined by local standards and geography. If 0.0 is entered for this value, the default value of 0.61 m/s is assumed.

h) DOMESTIC SEWER MAXIMUM VELOCITY (m/s) - the maximum allowable velocity in any domestic sewer pipe. This value is determined by local standards and geography. If 0.0 is entered for this value, the default value of 2.75 m/s is assumed.

i) STORM SEWER COVER DEPTH (m) - the minimum amount of ground cover for storm sewer pipes. If 0.0 is entered for this value, the default value of 3.35 m is assumed.

j) STORM SEWER MINIMUM VELOCITY (m/s) - the minimum allowable velocity in any storm sewer pipe. This value is determined by local standards and geography. If 0.0 is entered for this value, the default value of 0.61 m/s is assumed.

k) STORM SEWER MAXIMUM VELOCITY (m/s) - the maximum allowable velocity in any storm sewer pipe. This value is determined by local standards and geography. If 0.0 is entered for this value, the default value of 3.65 m/s is assumed.

l) MANHOLE SPACING (m) - the maximum spacing between manholes on a street. If 0.0 is entered for this value, the default value of 100 m is assumed.

m) STREET LIGHT SPACING (m) - the maximum spacing between street lights on a street where street lights are required. If 0.0 is entered for this value, the default value of 70 m is assumed.

Compile the above data on the "RACAM CONSTANTS" data sheet (Section A.1.8). Enter the data into the economic data file in the same order as on the data sheet, and according to the following format:

```
POPULATION DENSITY (F8.2)
WATERMAIN MIN. VELOCITY (F8.2)
              .
              .
              .
```

the format code "F8.2" being described above. When this section of the data file is complete, it should look like:

```
****3.45
****0.70
      .
      .
      .
```

where the "*" characters indicate spaces.

x) ACCOUNT NAMES AND NUMBERS

This data describes the names and numbers of the accounts among which the subdivision costs are divided. The following is included in this data section:

a) ACCOUNT NAME - the name of the account to which a potion of the subdivision costs are allotted

b) ACCOUNT # - the number of the account corresponding to the account name above

Compile the data for this section on the "ACCOUNT NAMES AND NUMBERS" data sheet (Section A.1.9). Enter the data into the economic data file from the data sheet according to the following format:

ACCOUNT # (A12) ACCOUNT NAME (A35)

The format code "A12" indicates that the data entered in the ACCOUNT # column is alpha-numeric (includes both numbers and letters) and is twelve characters long, while the format code "A35" indicates that the data entered in the ACCOUNT NAME column is alpha-numeric and is thirty-five characters long.

When all account names and numbers have been entered, the flag value "-1" must be entered in both the "ACCOUNT #" and "ACCOUNT NAME" column. When complete, this section of the data file should look like:

```
123-456-7890********************CITY OF REGINA
*234-345-456***************SPIKE'S CONSTRUCTION
      .                                .
      .                                .
      .                                .
**********-1**********************************-1
```

where the "*" characters indicate spaces. Note that the account names, numbers and flags are placed as far to the right as possible in their respective areas.

xi) ACCOUNT PERCENTAGES BY SUBDIVISION ELEMENT:

This data describes the percentages of each of the 21 subdivision element allotted to each of the accounts specified in the previous section of the data file. The information for this section may be obtained from municipal engineering departments.

This section includes the following:

a) # OF ACCOUNTS FOR EACH ELEMENT - indicates the number of accounts to which the subdivision element costs are allocated (eg.There may be three accounts to which the costs for element #1 are allotted, so enter a "3" in this position.)

b) # OF EACH ACCOUNT - the number of the account contributing to the cost of the subdivision element. Note that this is not the account number but the number specifying the position of the account in the previous section. If, for example, one of the accounts to which the cost for an element are allocated was the third account entered in the "ACCOUNT NAMES AND NUMBERS" section above, the number for this account is "3".

c) PERCENTAGE - the percentage of the subdivision element allocated to the account specified above. Enter the percentage as a fraction of 1 (eg. 1% is entered as 0.1).

0.01

Compile the data for this section on the "ACCOUNT PERCENTAGES BY SUBDIVISION ELEMENT" data sheet (Section A.1.10). Enter the data into the economic data file in the same order as on the data sheet and according to the following format:

```
# OF ACCOUNTS FOR ELEMENT #1 (I10)
# OF FIRST ACCOUNT (I10)   PERCENTAGE (F10.2)
# OF SECOND ACCOUNT (I10)   PERCENTAGE (F10.2)
          .                          .
          .                          .
          .                          .
# OF ACCOUNTS FOR ELEMENT #2 (I10)
          .
          .
          .
# OF ACCOUNTS FOR ELEMENT#21 (I10)
          .
          .
          .
```

the format codes "I10" and "F10.2" as described previously. When this data section is completed, it should look like:

```
*********2                (for element #1)
*********1******0.20
*********4******0.15
*********3                (for element #2)
*********2******0.19
*********4 *****0.13
*********5******0.20
          .
          .
          .
(up to element #21)
```

where the "*" characters indicate spaces.

xii) MAINTENANCE COSTS BY SUBDIVISION ELEMENT

This data describes the maintenance-cost information for each of the 21 subdivision elements. This information can be obtained from municipal engineering departments. The following is included in this section:

a) COST - the unit maintenance cost for each subdivision element.

b) GRADIENT - the amount that the above maintenance cost will increase annually

c) INTEREST - the interest rate on the maintenance cost above

Compile the data for this section on the "MAINTENANCE COST BY SUBDIVISION AREA" data sheet (Section A.1.11). Enter the data into the economic data file in the same order as on the data sheet and according to the following format:

```
COST1(F10.2)  GRADIENT1(F10.2)  RATE1(F10.2)  LIFE1(I10)
    .                 .                .            .
    .                 .                .            .
    .                 .                .            .
COST21(F10.2) GRADIENT21(F10.2) RATE21(F10.2) LIFE21(I10),
```

the format codes "F10.2" and "I10" being explained above. When complete, this data section should look like:

```
*******10.******401.********8.*********6
*******13.******500.********7.*********5
        .        .         .         .
        .        .         .         .
        .        .         .         .
*******14.******476.********9.*********8   (21 st line)
```

where the "*" characters indicate spaces.

xiii) OPERATION COSTS BY SUBDIVISION ELEMENT

This data describes the operation cost information for each of the 21 subdivision elements. The following is included in this section:

a) COST - the unit operation cost for each subdivision element.

b) GRADIENT - the amount that the above operation cost will increase annually

c) INTEREST - the interest rate on the operation cost above

Compile the data for this section on the "OPERATION COST BY SUBDIVISION ELEMENT" data sheet (Section A.1.12). Enter the data in the economic data file in the same order as on the data sheet and according to the following format:

```
COST1(F10.2)  GRADIENT1(F10.2)  RATE1(F10.2)  LIFE1(I10)
    .                .               .            .
    .                .               .            .
    .                .               .            .
COST21(F10.2) GRADIENT21(F10.2) RATE21(F10.2) LIFE21(I10),
```

the format codes "F10.2" and "I10" being explained above. When complete, this data section should look like:

```
********1.*******11.******8.77*********6
*******.85*******15.******7.63*********5
        .        .         .         .
        .        .         .         .
        .        .         .         .
*******.97*******16.******5.43*********8   (21 st line),
```

where the "*" characters indicate spaces.

2.2.2 LINK AND NODE DATA FILES:

CREATING A NEW LINK AND NODE DATA FILE

Before creating a new link and node data file, it is necessary to logon to the University of Regina's VAX 8600 computer, Meena (See "LOGIN PROCEDURE" above.), if you have not already done so.

ENTERING LINK AND NODE DATA

To begin entering link and node data, enter the following at the "$" prompt:

EDIT filename.DAT

Use a filename that can be easily remembered and relates to the area under consideration (eg. Use LINKREGNA.DAT as a filename for the link and node data file for Regina.)

After entering this command, the following will appear on the screen:

[EOB]

At the bottom of the screen will be the message:

Input file does not exist

The following link and node data can now be entered. (See Section C.2, "SAMPLE DATA FILES: LINK AND NODE DATA".)

i) NODE DATA

This data describes the nodes used for a subdivision plan. This data includes the following information entered in numerical order by node number:

a) NODE TYPE - defined by where the node is located. The different node types are:

1 - terminal (dead-end, cul-de-sac)

2 - directional (corners)

3 - three-way intersection

4 - four-way intersection

5 - other, including easements

Placing a minus sign ("-") in front of the node type indicates that the node is at an entrance or exit to the subdivision.

b) NODE ELEVATION (m)- the ground elevation at the location of the node. This elevation may be obtained from a contour map of the subdivision.

c) RADIUS (m) - the radius associated with a node. This is entered as a non-zero number only for nodes at bulb corners and cul-de-sacs. For all other cases, enter "0.0" for this entry.

In the case of bulb corners, the radius is measured from the center of curvature of the inside curve to the

outside of the bulb-corner, through the node. For a cul-de-sac, measure the radius from the node to the outside of the curved part of the cul-de-sac.

Compile the data for this section on the "NODE DATA" data sheet (Section A.2.1). Enter the data into the link and node data file in the order on the data sheet and according to the following format:

```
NODE TYPE(I10) ELEVATION(F10.2) RADIUS(F10.2) (for node 1)
      .               .               .       (for node 2)
      .               .               .
      .               .               .
```

the format codes "I10" and "F10.2" being described above. After all node data has been entered, enter the flag values "99","99." and "99." for the node type, elevation and radius respectively (note the placement of decimal points). When completed, this data section should look like:

```
*********2******25.2*******5.2
*********4******45.8*******0.0
         .       .          .
         .       .          .
         .       .          .
********99*******99.*******99.
```

where the "*" characters indicate spaces.

ii) LINK DATA I

This data describes the links used for a subdivision concept plan. This section of the link and node data file includes the following entered in numerical by link number:

a) ORIGIN NODE - the number of the node at one end of the link

b) DESTINATION NODE - the number of the node at the other end of the link

c) LINK LENGTH - the distance between the origin and destination node for the link

d) SINGLE-FAMILY UNITS - the number of single-family units on the current link. This information should be part of the subdivision concept plan.

e) MULTI-FAMILY UNITS - the number of multi-family units on the current link. This information should be part of the subdivision concept plan.

f) DRAINAGE AREA - the total area which drains into the current link. This can be determined from a contour map of the subdivision.

g) FRONTAGE - the total saleable frontage on the current link, including land for single- and multi-family dwellings. The saleable frontage is measured 6m from the street, except in the case of reverse-pie shaped lots where it is measured 18 m from the front of the lot. Frontage data can be obtained from a concept plan

of the subdivision, and both sides of the street must be considered for each link.

Compile the data for this section on the "LINK DATA I" data sheet (Section A.2.2). Enter the data into the link and node data file in the same order as the data sheet and according to the following format:

```
ORG DST(I10) LNG SGL MLT DRNG FRTG(F10.2)
   .          .   .   .    .    .
   .          .   .   .    .    .
   .          .   .   .    .    .
```

where:

ORG = ORIGIN NODE

DST = DESTINATION NODE

LNG = LINK LENGTH

MLT = MULTI-FAMILY UNITS

DRNG = DRAINAGE AREA

FRTG = FRONTAGE

The format codes "I10" and "F10.2" are described above.

After all the link data has been entered, enter the flag value "-1" for the origin and destination nodes, and "-1." (note decimal) for the remainder of the data. When complete, this data section should look like:

where the "*" characters indicate spaces.

iii) LINK DATA II

This data describes the road information associated with each link. It includes the following information, entered in numerical order by link number:

a) ROAD LENGTH (m) - the length of the road for each link. In most cases, the road length should be the same length as the link length. If there is no road on a link, the road length should be entered as "0.0".

b) ROAD WIDTH (m) - the width of the road for each link. If there is no road on a link, the road width should be entered as "0.0".

c) ROAD TYPE - the type of road on each link. A road may be one of two types - collector or local. Enter a "1" for a collector street or "2" for a local street.

Compile the data for this section on the "LINK DATA II" data sheet (Section A.2.3). Enter the data in the same order as on the data sheet and according to the following format:

ROAD LENGTH(F10.2) ROAD WIDTH(F10.2) ROAD TYPE(I10)

the format codes "F10.2" and "I10" being described above. When complete, this data section should look like:

```
****135.64*****26.00*********2     (for link #1)
****123.45*****10.00*********1     (for link #2)
       .          .           .
       .          .           .
       .          .           .
```

where the "*" characters indicate spaces.

iv) LINK DATA III

This data describes the sidewalk information associated with each link. It includes the following:

a) SIDEWALK WIDTH (m)- the width of the sidewalk associated with the link. If there is no sidewalk on the link, enter "0.0" for this value.

b) BARRIER CURB AND GUTTER LENGTH (m) - the length of barrier curb and gutter on the link. If there is no barrier curb and gutter on the link, enter "0.0" for this value.

c) ROLLED CURB AND GUTTER LENGTH (m) - the length of rolled curb and gutter on the link. If there is no rolled curb and gutter on the link, enter "0.0" for this value.

d) BOULEVARD CURB AND GUTTER LENGTH (m) - the length of boulevard curb and gutter on the link. If there is no boulevard curb and gutter on the link, enter "0.0" for this value.

Compile the data for this section on the "LINK DATA III" data sheet (Section A.2.4). Enter the data in the same order as on the data sheet and according to the following format:

WIDTH(F10.1)	BARRIER(F10.1)	ROLLED(F10.1)	BLVD(F10.1)
.	.	.	.
.	.	.	.
.	.	.	.

the format code "F10.1" indicating that the data in each of the four columns is read as ten digits long (including decimal points and spaces) and one decimal place is assumed if no decimal point is entered. When complete, this data section should look like:

```
*******1.6****126.00*****106.7*******9.1     (for link #1)
******1.45****110.00*****100.1*******0.0     (for link #2)
       .          .           .
       .          .           .
       .          .           .
                          .
```

where the "*" characters indicate spaces.

v) LINK DATA IV

This data describes initial population and area information required for domestic and storm sewer calculations. It includes the following:

LINK # - the number of the link which requires initial population or area information

WATERMAIN POPULATION - the initial population of the link for watermain calculations. This population consists of people that are not considered part of the residential sector who are serviced by the watermain for the link. This population includes people using a school or working in a commercial complex within the study area, as well as people located outside the study area being serviced by a watermain within the study area.

DOMESTIC SEWER POPULATION - the initial population of the link for domestic sewer calculations. This population includes people that are not considered part of the residential sector (as described above for "WATERMAIN POPULATION") but are serviced by a domestic sewer pipe within the study area.

STORM SEWER AREA (Ha.) - the initial drainage area of the link for storm sewer calculations. This area consists of land that is not considered part of the residential sector which is drained by the storm sewer pipe for the

link. This area includes large parking lots, such as those near schools and commercial areas, and land areas outside the study area that are serviced by a storm sewer within the study area.

Compile the data for this section on the "LINK DATA IV" data sheet (Section A.2.5). Enter the data into the link and node data file from the data sheet according to the following format:

```
#(I10)  WPOP(F10.2)  DSPOP(F10.2)  SSAREA(F10.2)
  .         .              .             .
  .         .              .             .
  .         .              .             .
```

the format codes "I10" and "F10.2" being described above. After all the link data for this section has been entered, the flag value "-1" should be entered in the "LINK #" column and the flag "-1." should be entered in the other three columns (Note the decimal point) When complete, this data section should look like:

```
********11*******1.0*******0.0*******0.0
        .        .         .         .
        .        .         .         .
        .        .         .         .
        -1       -1.       -1.       -1.
```

where the "*" characters indicate spaces. Note that this data need only be entered for links where initial values are required for calculations.

vi) SUBDIVISION AREAS

This data consists of the areas for the various subdivision sections. It includes the following:

a) PARK AREA (Ha.)- the total park area in the subdivision
b) BUFFER AREA (Ha.) - the total buffer area in the subdivision. This includes areas which provide a buffer between lots and streets as well as public land not considered park land.
c) SCHOOL AREA (Ha.) - the total subdivision area devoted to schools
d) MULTI-FAMILY DWELLING AREA (Ha.) - the total subdivision area devoted to multi-family units
e) COMMERCIAL AREA (Ha.) - the total commercial area in the subdivision
f) OTHER (Ha.) - includes all land not accounted for in the above categories
g) TOTAL SUBDIVISION AREA (Ha.) - the total area of all land in the subdivision

Compile the data for this section on the "SUBDIVISION AREAS" data sheet (Section A.2.6). Enter the data in the same order as on the data sheet and according to the following format:

```
PARK AREA (F10.2)
BUFFER AREA (F10.2)
      .
      .
      .
```

the format code "F10.2" being described above. After all the area information has been entered, enter the flag value "-1". When this section is complete, it should look like:

```
******7.37
******0.27
       .
       .
       .
********-1
```

where the "*" characters indicate spaces.

CHAPTER 3

RACAM OPERATION

3.1 RUNNING RACAM

Before running RACAM, it is necessary to logon to the University of Regina's VAX 8600 computer, Meena, from a VT-100 terminal. (See Section 2.1.1, "ECONOMIC DATA FILES: LOGIN PROCEDURE".)

The RACAM program requires execution of the main RACAM routine, RCMAIN. Enter the following at the dollar sign prompt to do so:

RUN RCMAIN <RETURN>

The RACAM heading will appear on the screen. Press <RETURN> to continue.

The next prompt to appear is:

Please, enter the name of the OUTPUT file:

Enter the name of the filename to where the RACAM output is to be written, then press <RETURN>. Enter the filename in the form "filename.dat". The extension (the last three letters following the period) "dat" indicates that the file where the RACAM output is to be written is a data file. If the extension is omitted, and the filename is entered in the

form "filename", the ".dat" extension is assumed. "Filename" can be up to 39 characters long.

The following prompt is:

E C O N O M I C D A T A

Please, enter the name of the INPUT file:

Enter the name of the file containing the economic data for the area being considered. Although this file is a data file, and thus has the extension "dat", it is not necessary to include this extension when entering the filename. This file must have been previously created for RACAM to continue execution. If the data file has not been created, RACAM will respond with:

F I L E N O T F O U N D

FORTRAN STOP

If an error has been made while entering the filename, re-execute RACAM by typing the following command:

RUN RCMAIN <RETURN>

Otherwise, create an economic data file before executing RACAM again. (See Section 2.2.1,"ECONOMIC DATA FILES: CREATING A NEW ECONOMIC DATA FILE".)

After RACAM has completed reading the economic data file, the following prompt will appear:

L I N K / N O D E D A T A

Please, enter the name of the INPUT file:

Enter the name of the link/node data file for the study area. Like the economic data file, the link/node data file must be created before executing RACAM. Otherwise the message:

F I L E N O T F O U N D

FORTRAN STOP

will appear. If an error is made while entering the data file name, re-execute RACAM, otherwise create the required link/node data file. (See Section 2.2.2,"LINK AND NODE DATA FILES: CREATING A NEW LINK AND NODE DATA FILE".)

RACAM will print a list of default values which are used in calculations and then show the message:

PROCESSING.....

This message indicates that RACAM is analyzing the subdivision plan. The analysis requires approximately 30 seconds for a subdivision of 70 nodes. but this time varies, depending upon depending on the size of the subdivision (number of nodes and links), and how many other users are accessing the VAX 8600. After the calculations have been completed, RACAM returns a dollar sign ($) indicating that program execution has been successfully completed.

3.2 RACAM DIAGNOSTIC MESSAGES

During program execution, RACAM indicates the occurrence of errors and provides information regarding operation of RACAM using diagnostic messages. These diagnostic messages may be written to the screen during RACAM execution, or to a file titled "ERRORFILE.DAT".

There are two types of diagnostic messages. The first, TYPE 1, is a statement appearing on the screen which states that an error has occurred, or indicates that RACAM is operating normally. The second type of diagnostic message, TYPE 2, which is written to the file "ERRORFILE.DAT", appears in the form:

mmmmm...-n-iiiiii....

where:

1. "mmmmm..." is a module (subroutine) name indicating where the message originates.
2. "n" is a letter indicating the type of message. The message may be one of three types:
   a) E - indicating an unrecoverable error has occurred.
   b) I - indicating an informational message only.

c) W - indicating a warning. An error has occurred but the program will try to continue processing.

3."iiiiii..." is additional information regarding the diagnostic message.

3.2.1 TYPE 1 MESSAGES

a) Message: FILE INCOMPLETE

Routines: INDAT1, INDAT2

Explanation: This message appears on the screen after an input filename has been entered. It indicates RACAM input routines have not detected the end-of-file flag for the input file last specified (link and node, or economic data file). Either the data file has not been completed, or the end-of-file flag was not entered.

Action: If the data file last specified before the message appeared is incomplete, enter the remainder of the data, ensuring that the file is terminated with the appropriate end-of-file flags. If the file simply is not terminated with the appropriate end-of-file flags, add these flags to the end of the file. (See page 17, "CREATING DATA FILES" for information on end-of-file flags and editing a data file.)

b) Message: F I L E N O T F O U N D
FORTRAN STOP

Routine: IOFILE

Explanation: This message appears on the screen after an input filename has been entered. It indicates that RACAM could not find the specified file for data input. The "FORTRAN STOP" portion of the message indicates that program execution is stopped in this case.

Action: If a spelling mistake or typographical error was made while entering the name of the input file, execute RACAM from the beginning, assuring that the filenames are spelled correctly while entering them (See page 41, "Running RACAM"). If, however, no mistakes were made in the filename, the file does not exist and must be created before executing RACAM using that particular filename as an input file.

c) Message: COMPLETED...

Routine: INDAT1,INDAT2

Explanation: This message appears on the screen after an input filename has been entered. It indicates that the data file specified has been successfully read. This message may appear on the screen for only a short period (one second or less).

d) Message: PROCESSING...

Routine: RCMAIN
Explanation: This message appears on the screen after the output filename and the input filenames have been entered. It indicates that RACAM is doing calculations and writing information to the output file.

3.2.2 TYPE 2 MESSAGES

a) Message: CATBAS-W-No drainage (flat surface for link # xx)

Routine: CATBAS

Explanation: This message is written to ERRORFILE.DAT if both origin and destination nodes for a given link are at the same elevation.

b) Message: CATBAS-W-No drainage at exit node # xx

Routine: CATBAS

Explanation: This message is written to ERRORFILE.DAT if all the nodes connected to a subdivision exit node by links are of lower elevation than the exit node.

Action: No user actions are necessary. RACAM will make appropriate adjustments in its calculations to account for the higher elevation of the exit node. Cutting and filling may be done by the program in this case.

c) Message: ERRCHK-I-Link # xx has the same node at each end

Routine: ERRCHK

Explanation: This message is written to ERRORFILE.DAT if the two node numbers entered for a link in the link and node

data file are the same. This condition is improbable, since it indicates a link that does not go anywhere.

Action: If, in this case, a link that has the same nodes at each end is not intended, enter the appropriate node numbers in the link and node data file for the link specified. (See page 17, "Creating Data Files")

d) Message: CREATE-I-Link # xx has no path to an exit.

Routine: CREATE

Explanation: This message indicates that the link specified is not connected to an exit node, either directly or through other links.

Action: If the specified link is supposed to be connected to an exit node, there is an error in the link and node data file. Check to make sure that all the link and node data has been entered correctly and make the appropriate corrections. (See page 17, "Creating Data Files")

e) Message: CREATE-I-A path choice was made in domestic sewer system:

- Link not chosen - # xx
- Link chosen - # yy
- Location (node) - # zz

Routine: CREATE

Explanation: This message indicates that one link was chosen over another during the domestic sewer design procedure. The node at which this choice took place is also indicated.

f) Message: CREATE-I-A path choice was made in storm sewer system:

- Link not chosen - # xx
- Link chosen - # yy
- Location (node) - # zz

Routine: CREATE

Explanation: This message indicates that one link was chosen over another during the storm sewer design procedure. The node at which this choice took place is also indicated.

CHAPTER 4

RACAM OUTPUT

4.1 VIEWING RACAM OUTPUT

All data output during RACAM execution is written to the output data file specified at the prompt:

Please, input the name of the INPUT file:

This RACAM output is written in tabular and graphical format.

Two methods of viewing a RACAM output data file exist. The first method uses the VAX/VMS **TYPE** command. At the "$" prompt, enter command **TYPE**, followed by the RACAM output filename, "filename.dat". Note that the extension "dat" is required. The RACAM output file will be printed on the screen which will quickly scroll. To halt the scrolling, press <Ctrl-S>. (Press the "Ctrl" key then press the "S" key.) To resume scrolling, press <Ctrl-Q>.

The second method to view the RACAM output data file is to obtain a hard-copy (printout). Enter the command **PRINT** at the "$" prompt, followed by "filename.dat". (Note: variations of the **PRINT** command may be used depending on the configuration of the system in use - see VAX manuals for more information on the **PRINT** command.) Specify the "dat" extension for this command. This **PRINT** command will send the RACAM output data file to the default printer. RACAM will generate approximately 60 pages of output for a subdivision with 70 nodes, the number of pages increasing with the size of the subdivision.

4.2 OUTPUT DESCRIPTION

The RACAM output data file includes the RACAM analysis information, and some of the original information from the input data files. The information written to the RACAM output data file includes:

4.2.1 Unit Costs

This first section of the output data file comprises the unit costs for the subdivision. These are the same unit costs that are entered in the economic data input file. This unit cost output data can be compared with the original unit cost input data to check for errors in the economic input data file. For an explanation of the unit cost data elements, refer to the section listed with each element below.

The output for this section appears in the following format:

DESCRIPTION (Subdivision element or service)	UNITS	UNIT PRICE

The unit cost data for subdivision elements or services is output in the following order:

1. STREET SIGN ($/INTERSECTION) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)
2. STREET LIGHT ($/LIGHT) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)

3. COLLECTOR STREET ($/m^2) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)
4. LOCAL STREET ($/m^2) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)
5. SIDEWALK ($/m^2) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)
6. BARRIER CURB AND GUTTER ($/m) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)
7. ROLLED CURB AND GUTTER ($/m) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)
8. BOULEVARD CURB AND GUTTER ($/m) - Section 2.2.1, ENTERING ECONOMIC DATA, (ii)
9. PARK DEVELOPMENT ($/Ha) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
10. BUFFER DEVELOPMENT ($/Ha) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
11. POWER: SINGLE-FAMILY DWELLING UNIT ($/UNIT) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
12. POWER: MULTI-FAMILY DWELLING UNIT ($/UNIT) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
13. GAS: SINGLE-FAMILY DWELLING UNIT ($/UNIT) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
14. GAS: MULTI-FAMILY DWELLING UNIT ($/UNIT) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
15. PHONE: SINGLE-FAMILY DWELLING UNIT ($/UNIT) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)

16. PHONE: MULTI-FAMILY DWELLING UNIT ($/UNIT) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
17. STREET SWEEPING ($/KM/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
18. WINTER ROAD MAINTENANCE ($/KM/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
19. ASPHALT MAINTENANCE ($/KM/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
20. CONCRETE MAINTENANCE ($/KM/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
21. SEWER MAINTENANCE ($/KM/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
22. WATER MAINTENANCE ($/KM/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
23. SOLID WASTE COLLECTION ($/HA/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
24. STREET LIGHT MAINTENANCE ($/LIGHT/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
25. PARK AND BUFFER MAINTENANCE ($/HA/YR) - Section 2.2.1, ENTERING ECONOMIC DATA, (viii)
26. CATCHBASIN COSTS:
    - i) BASE,FRAME,COVER ($/CATCHBASIN) - Section 2.2.1, ENTERING ECONOMIC DATA, (vi)
    - ii) SHAFT ($/M-DEPTH) - Section 2.2.1, ENTERING ECONOMIC DATA, (vi)
    - iii) LEAD ($/M-9 Depth Indexes) - Section 2.2.1, ENTERING ECONOMIC DATA, (vii)

27. MANHOLE COSTS:

    i) BASE,FRAME,COVER ($/CATCHBASIN) - Section 2.2.1, ENTERING ECONOMIC DATA, (vi)

    ii) SHAFT ($/M-DEPTH) - Section 2.2.1, ENTERING ECONOMIC DATA, (vi)

28. WATERMAIN COSTS @ 2.44 METERS OF COVER ($/M for 6 Diameter Indexes) - Section 2.2.1, ENTERING ECONOMIC DATA, (iii)

29. DOMESTIC SEWER ($/M for 9 Depth Indexes, 5 Diameter indexes.) - Section 2.2.1, ENTERING ECONOMIC DATA, (iv)

30. STORM SEWER ($/M for 9 Depth Indexes, 14 Diameter Indexes.) - Section 2.2.1, ENTERING ECONOMIC DATA, (v)

4.2.2 LINK DATA

This second section of the output data file includes all the input data for each of the links used to describe the subdivision. The output for this section can be checked against the original input data to check for errors in the link and node input data file. For an explanation of the link data elements, refer to the section listed with each element below.

The following information for each link is included in this section:

1. LINK NUMBER - Section 2.2.2, ENTERING LINK AND NODE DATA, (ii)

2. ORIGIN NODE (labelled as NODE1) - Section 2.2.2, ENTERING LINK AND NODE DATA, (ii)
3. DESTINATION NODE (labelled as NODE2) - Section 2.2.2, ENTERING LINK AND NODE DATA, (ii)
4. LINK LENGTH (m) - Section 2.2.2, ENTERING LINK AND NODE DATA, (ii)
5. NUMBER OF SINGLE-FAMILY DWELLING UNITS - Section 2.2.2, ENTERING LINK AND NODE DATA, (ii)
6. NUMBER OF MULTI-FAMILY DWELLING UNITS - Section 2.2.2, ENTERING LINK AND NODE DATA, (ii)
7. STREET LENGTH FOR LINK (m) - Section 2.2.2, ENTERING LINK AND NODE DATA, (iii)
8. STREET WIDTH FOR LINK (m) - Section 2.2.2, ENTERING LINK AND NODE DATA, (iii)
9. STREET TYPE FOR LINK (Collector or Local) - Section 2.2.2, ENTERING LINK AND NODE DATA, (iii)
10. SIDEWALK WIDTH FOR LINK (m) - Section 2.2.2, ENTERING LINK AND NODE DATA, (iv)
11. BARRIER CURB AND GUTTER LENGTH FOR LINK (m) - Section 2.2.2, ENTERING LINK AND NODE DATA, (iv)
12. ROLLED CURB AND GUTTER LENGTH FOR LINK (m) - Section 2.2.2, ENTERING LINK AND NODE DATA, (iv)
13. BOULEVARD CURB AND GUTTER LENGTH FOR LINK (m) - Section 2.2.2, ENTERING LINK AND NODE DATA, (iv)

4.2.3 NODE DATA

This section of the output file includes information about the nodes used to describe the subdivision. The output for this section can be compared with the original node data to check for errors in the link and node input data file. For an explanation of the node data elements, refer to the section listed with each element below.

The following information for each node is included in this section:

1. NODE NUMBER - Section 1.1.2 (xiii)
2. NODE TYPE, where:

   1 = Termination (Cul-de-Sac, Dead-End)

   2 = Directional (Corner)

   3 = 3-Way Street Intersection

   4 = 4-Way Street Intersection

   5 = Other (eg. Easements)

   - Section 1.1.2 (xiii)
3. NODE ELEVATION (m) - Section 1.1.2 (xiii)
4. RADIUS ASSOCIATED WITH NODE (m) - Section 1.1.2 (xiii)

4.2.4 CALIBRATION DATA VALUES

This section of the output data file contains the RACAM constants given default values. Only certain variables, having been assigned to 0 in the economic data input file, are given default values. These default values are displayed on the terminal screen during program execution as well as being included in the output data file. The default values are averages which are generally applicable, and have been obtained by considering values for several different areas. For an explanation of each calibration data elements, refer to Section 2.2.1 (ix).

Depending on which values are assigned to 0 in the economic data file, this calibration data may include:

1. Estimated Number of People Per Unit
2. Estimated Number of People Per Acre
3. Minimum Domestic Sewer Pipe Velocity (m/s)
4. Minimum Storm Sewer Pipe Velocity (m/s)
5. Maximum Domestic Sewer Pipe Velocity (m/s)
6. Maximum Storm Sewer Pipe Velocity (m/s)
7. Minimum Watermain Pipe Velocity (m/s)
8. Manning's Roughness Coefficient (Unitless)
9. Minimum Cover Depth for Domestic Sewer (m)
10. Minimum Cover Depth for Storm Sewer (m)
11. Maximum Spacing Between Manholes (m)
12. Maximum Spacing Between Lights (m)

13. Expected Domestic Water Use (Gallons/Minute)

14. Water Consumption Safety Factor (Unitless)

4.2.5 WATERMAIN TRIBUTARY DATA

This section of the output data file includes the tributary data for the subdivision watermain system. The information output in this section includes:

1. Feed Links - numbers of the links whose watermain pipes provide flow into a certain node. There can be up to three feed links at a node.
2. Receptor Link - number of the link whose watemain pipe is providing flow out of a certain node. There can only be one receptor link at a node.

The feed and receptor links on one link of the tributary data section all have a common node. If the following were an entry in the watermain tributary data:

12 19 0 - 10

with 12, 19 and 0 in the "Feed Links" column and 10 in the "Receptor Links" column, the links 12, 19 and 10 would be connected together at a node. Although the node number is not specified in the watermain tributary data section, it could be determined by finding the node where links 12, 19 and 10 connect on the original subdivision concept plan.

As well as indicating that the links are connected to a common node, the sample entry above indicates that the watermain pipes along links 12 and 19 are flowing into the common node (or feeding the node) while the watermain pipe along node 10 is flowing out of the common node. The "0" in the "Feed Links" column indicates that there is no third feed link.

4.2.6 DOMESTIC SEWER TRIBUTARY DATA

This section of the output data file includes the tributary data for the subdivision domestic sewer system. The information output in this section includes:

1. Feed Links - numbers of the links whose domestic sewer pipes provide flow into a certain node. There can be up to three feed links at a node.
2. Receptor Link - the number of the link whose domestic sewer pipe provides flow out of a certain node.

The format of the output for this section and its meaning is the same as for the "Watermain Tributary Data" section, except that domestic sewer flow is considered. See the example in the "Watermain Tributary Data Section".

4.2.7 STORM SEWER TRIBUTARY DATA

This section of the output data file includes the tributary data for the subdivision storm domestic sewer system. The information output in this section includes:

1. Feed Links - the numbers of the links whose storm sewer pipes provide flow into a certain node. There can be up to three feed links at a node.
2. Receptor Link - the number of the link whose storm sewer pipe provides flow out of a certain node. There can only be one receptor link at a node.

The format of the output for this section and its meaning is the same as for the "Watermain Tributary Data" section, except that storm sewer flow is considered. See the example in the "Watermain Tributary Data Section".

4.2.8 WATERMAIN LINK DATA

This section of the output data file includes watermain information for the links in the subdivision. This watermain information includes:

1. LINK # - Note that they are output in the order they are calculated by the program, not in numerical order.
2. TRIBUTARY POPULATION - all people with access to the watermain for the link listed. Included in this population are those people with access to links into which the current link flows
3. FLOW RATE (m^3/s) - the rate at which the water flows in the watermain along the given link in order to supply the tributary population.
4. REQUIRED DIAMETER (m) - the minimum diameter of watermain pipe allowing water to flow at the given flow rate.

4.2.9 DOMESTIC SEWER LINK DATA

This section of the output data file describes the domestic sewer information for each link where domestic sewer pipes are present. First, however, minimum and maximum domestic sewer calibration values are output (See Section 2.2.1 (ix)):

1. MINIMUM ACCEPTABLE PIPE VELOCITY (m/s)
2. MAXIMUM ACCEPTABLE PIPE VELOCITY (m/s)
3. MINIMUM DEPTH OF GROUND COVER OVER PIPE (m)

The domestic sewer data output for each of the links includes:

1. LINK # - Note that these are not in numerical order.
2. TRIBUTARY POPULATION - all people with access to the domestic sewer for the link listed. Included in this population are those people with access to links from which the current link flows
3. FLOW RATE (m^3/s) - the rate at which the water flows in the domestic sewer along the given link in order to supply the tributary population.
4. GROUND SLOPE (m/m) - the change in vertical elevation between the terminal nodes divided by the link length.
5. CALCULATED DIAMETER (m) - the minimum diameter of the pipe required to carry the calculated flow.
6. VELOCITY (m) - the speed of the flow in the pipe for the calculated flow rate and the pipe diameter.

7. PIPE SLOPE (m/m) - the vertical change in elevation of the sewer pipe per unit length of the pipe. This is different from GROUND SLOPE (above) due to excavation.
3. DEPTH OF EXCAVATION (m) - This indicates the depth of the sewer pipe bedding from the ground elevation at the two ends of the pipe (NODE1 and NODE2).

4.2.10 STORM SEWER LINK DATA

This section of the output data file describes the storm sewer information for each link where storm sewer pipes are present. First, however, minimum and maximum storm sewer calibration values are output (See Section 2.2.1 (ix)):

1. MINIMUM ACCEPTABLE PIPE VELOCITY (m/s)
2. MAXIMUM ACCEPTABLE PIPE VELOCITY (m/s)
3. MINIMUM DEPTH OF GROUND COVER OVER PIPE (m)

The storm sewer data output for each of the links includes:

1. LINK # - Note that these are not in numerical order.
2. TRIBUTARY POPULATION - all people with access to the storm sewer for the link listed. This includes those people with access to links which flow into the current link.
3. FLOW RATE (m^3/S) - the rate at which the water will flow in the pipe according to RACAM storm sewer design and analysis results.
4. GROUND SLOPE (m/m) - the change in vertical elevation between the terminal nodes of the link divided by the length of the link.
5. CALCULATED DIAMETER (m) - the minimum diameter of the pipe required to carry the calculated flow.
6. VELOCITY (m) - The speed of the flow in the pipe for the calculated flow rate and the pipe diameter.

7. PIPE SLOPE (m/m) - the vertical change in elevation of the sewer pipe per unit length of pipe. This may be different from GROUND SLOPE (above) due to excavation.
8. DEPTH OF EXCAVATION (m) - indicates the depth of the sewer pipe bedding from ground elevation at the two ends of the pipe (NODE1 and NODE2).

4.2.11 STORM SEWER SYSTEM DATA

This section of the output data file summarizes the RACAM storm sewer calculations. The data appears in a tabular format which includes the following for each link:

1. LINK # - includes all subdivision links in numerical order.
2. PIPE LENGTH (m) - equivalent to the length of the link. A pipe length of zero indicates that there is no storm sewer pipe along the current link.
3. DIAMETER (mm) - The storm sewer pipe diameter originally calculated is rounded up to the nearest even pipe diameter to obtain this value.
4. UNIT COST ($/m) - the unit cost for the particular diameter of pipe used for the link.
5. TOTAL COST - the total cost of the pipe for the link. It is obtained by multiplying UNIT COST by PIPE LENGTH.

4.2.12 DOMESTIC SEWER SYSTEM DATA

This section of the output data file summarizes the RACAM domestic sewer calculations. The data appears in a tabular format which includes the following for each link:

1. LINK # - includes all subdivision links listed in numerical order.
2. PIPE LENGTH (m) - equivalent to the length of the link. A pipe length of zero indicates that there is no domestic sewer pipe along the current link.
3. DIAMETER (mm) - The domestic sewer pipe diameter originally calculated is rounded up to the nearest even pipe diameter to obtain this value.
4. UNIT COST ($/m) - the unit cost for the particular diameter of pipe used for the link.
5. TOTAL COST - the total cost of the pipe for the link. It is obtained by multiplying UNIT COST by PIPE LENGTH.

4.2.13 WATERMAIN SYSTEM DATA

This section of the output data file summarizes the RACAM watermain calculations. The data appears in a tabular format which includes the following for each link:

1. LINK # - includes all subdivision links listed in numerical order.
2. PIPE LENGTH (m) - equivalent to the length of the link. A pipe length of zero indicates that there is no watermain pipe along the current link.
3. DIAMETER (mm) - The watermain pipe diameter originally calculated is rounded up to the nearest even pipe diameter to obtain this value.
4. UNIT COST ($/m) - the unit cost for the particular diameter of pipe used for the link.
5. TOTAL COST - the total cost of the pipe for the link. It is obtained by multiplying UNIT COST by PIPE LENGTH.

4.2.14 MANHOLES AND CATCHBASINS

This section of the output data file summarizes the manhole and catchbasin information for each of the links. This data section includes:

1. LINK # - includes all subdivision links listed in numerical order.
2. DOMESTIC SEWER MANHOLES:
   a) QUANTITY - the number of domestic sewer manholes on the link
   b) COST - the total cost for domestic sewer manholes on the link
3. STORM SEWER MANHOLES:
   a) QUANTITY - the number of storm sewer manholes on the link
   b) COST - the total cost for all the storm sewer manholes on the link
4. STORM SEWER CATCHBASINS:
   a) QUANTITY - the number of storm sewer catchbasins on the link
   b) COST - the total cost for all the storm sewer catchbasins on the link

The quantities and costs for each of the three manhole or catchbasin sections are totalled at the bottom of the table.

4.2.15 STREET REQUIREMENTS

This section describes quantities and costs of street lights, street signs and traffic signs for each link in the subdivision. The following information is included in this section:

1. LINK # - includes all subdivision links listed in numerical order.
2. STREET LIGHTS:
   - QUANTITY - the number of street lights on each link
   - COST - the total cost for street lights on the link
3. STREET SIGNS:
   - QUANTITY - the number of street signs on each link
   - COST - the total cost for street signs on the link
3. TRAFFIC SIGNS:
   - QUANTITY - the number of traffic signs on each link
   - COST - The total cost for traffic signs on the link.

The quantities and costs for each of the above three street requirement sections are totalled at the bottom of the table.

4.2.16 PAVEMENT INFORMATION

This section of the output data file describes the collector and local streets for each of the links in the subdivision. The following information, presented in tabular form, is included in this section:

1. LINK # - includes all subdivision links listed in numerical order.
2. LOCAL STREET:

   LENGTH (m) - the length of the local street for the link. If a length of zero appears in this column, this indicates that there is no local street on the link.

   AREA (m^2) - the local street pavement area associated with the link
3. COLLECTOR STREET:

   LENGTH (m) - the length of the collector street for the link. If a length of zero appears in this column, this indicates that there is no collector street on the link.

   AREA (m^2) - the collector street pavement area associated with the link
4. TOTAL:

   LENGTH (m) - the total length of both collector and local streets for the link

AREA (m^2) - the total pavement area of both collector and local streets for the link

COST - the total cost of both collector and local streets for the link

Since links will have only one type of street on them (either collector or local), the values in the TOTAL section will be the same as those in either the LOCAL STREET or COLLECTOR STREET section.

Totals for each of the sections described above appear at the bottom of the table, indicating street lengths, areas and costs for the subdivision.

4.2.17 SIDEWALK REQUIREMENTS

This section of the output data file describes sidewalk areas, curb lengths, and curb and sidewalk costs for each link in the subdivision. For an explanation of the curb and gutter elements, refer to the sections listed with the elements below.

This data section, which is in tabular form, includes the following for each link:

1. LINK # - includes all subdivision links listed in numerical order.
2. BARRIER CURB AND GUTTER LENGTH (m) - Section 1.1.2 (xvi)
3. ROLLED CURB AND GUTTER LENGTH (m) - Section 1.1.2 (xvi)
4. BOULEVARD CURB AND GUTTER LENGTH (m) - Section 1.1.2 (xvi)
5. SIDEWALK AREA (m^2) - the area for the sidewalk along the link
6. TOTAL COSTS - includes the costs for the above four elements

The quantities and costs for each of the above sidewalk requirement sections are totalled at the bottom of the table.

4.2.18 FAMILY UNITS

This section of the output data file consists of information about single- and multi- family dwelling units for each link in the subdivision. This data section, which is in tabular form, includes the following for each link:

1. SINGLE-FAMILY UNITS:
   a) NUMBER OF UNITS - the number of single-family dwelling units on the link
   b) GAS COSTS - the cost of constructing gas lines for each single-family dwelling on the current link
   c) POWER COSTS - the cost of constructing power lines for each single-family dwelling on the current link
   d) TELEPHONE COSTS - the cost of constructing telephone lines for each single-family unit on the current link
2. MULTI-FAMILY UNITS:
   (Same sub-headings are used as for single-family units above, except referring to multi-family units.)

The quantities and costs for each of the above sections are totalled at the bottom of the table.

4.2.19 CONSTRUCTION COSTS

This section of the output data file describes several unit construction costs for each link in the subdivision. This data section, which is in tabular form, includes the following for each link:

1. TOTAL CONSTRUCTION COST - includes all construction costs associated with the link
2. CONSTRUCTION COST PER PAVEMENT AREA - the total construction cost divided by the link pavement area.
3. CONSTRUCTION COST PER STREET LENGTH - the total construction cost divided by the street length associated with the link
4. CONSTRUCTION COST PER SALEABLE FRONTAGE - the total construction cost divided by the saleable frontage for the link
5. CONSTRUCTION COST PER DWELLING UNIT - the total construction cost divided by the number of dwelling units on the link

The total construction costs for each link in the CONSTRUCTION COST column are totalled at the bottom of the table, giving the total construction cost for the subdivision.

4.2.20 SUMMARY - CONSTRUCTION COSTS

This section of the output data file summarizes the construction costs for the subdivision. The information for this section is output in the following format:

DESCRIPTION (Name of element)	QUANTITY	COST

The data for this section is output for each subdivision element in the following order:

1. WATERMAIN
2. DOMESTIC SEWER
3. STORM SEWER
4. COLLECTOR STREET
5. LOCAL STREET
6. SIDEWALK
7. BARRIER CURB AND GUTTER
8. ROLLED CURB AND GUTTER
9. BOULEVARD CURB AND GUTTER
10 PARK DEVELOPMENT
11. BUFFER DEVELOPMENT
12. CATCHBASIN
13. DOMESTIC SEWER MANHOLE
14. STORM SEWER MANHOLE
15. STREET LIGHT
16. STREET SIGN

17. TRAFFIC SIGN
18. GAS: SINGLE-FAMILY UNIT
19. POWER: SINGLE-FAMILY UNIT
20. TELEPHONE: SINGLE-FAMILY UNIT
21. GAS: MULTI-FAMILY UNIT
22. POWER: MULTI-FAMILY UNIT
23. TELEPHONE: MULTI-FAMILY UNIT

4.2.21 SITE SPECIFIC INFORMATION

This section of the output data file gives a summary of the intersection type and the land use for the subdivision. The following is included in this data section:

1. NUMBER OF ROAD ELEMENTS:
    a) CUL-DE-SACS
    b) DEAD-ENDS
    c) THREE-WAY INTERSECTIONS
    d) FOUR-WAY INTERSECTIONS
    e) SUBDIVISION EXITS
2. LAND USE DISTRIBUTION (AREA IN Ha):
    a) NET SINGLE-FAMILY
    b) NET MULTI-FAMILY
    c) SCHOOL
    d) PARK
    e) BUFFER
    f) COMMERCIAL
    g) PAVEMENT
    h) SIDEWALK
    i) OTHER

At the bottom of this LAND USE DISTRIBUTION summary is a total of all the areas, giving GROSS SUBDIVISION AREA in Ha.

4.2.22 SUBDIVISION CONSTRUCTION COST

This section of the output data file describes various unit construction costs for the subdivision. For all unit costs below, a lower value indicates greater efficiency than a higher one.. This data section includes:

1. TOTAL CONSTRUCTION COST PER TOTAL SUBDIVISION AREA ($/Ha) - indicates the per unit cost to build the complete subdivision.
2. TOTAL CONSTRUCTION COST PER GROSS SUBDIVISION AREA ($/Ha) - Gross subdivision area includes residential land, road allowances, parks, playgrounds and schools, but does not include commercial areas. The total construction cost per gross area value can be compared for different proposed subdivisions to give an indication of which is more cost efficient, neglecting commercial areas.
3. TOTAL CONSTRUCTION COST PER NET SUBDIVISION AREA ($/Ha) - Net subdivision area is the area of land used entirely for dwellings sites. The total construction cost per net subdivision area value can be compared for different proposed subdivisions to indicate which is more cost effective considering dwelling units. This is important since the since the sale of land comprising the net subdivision area accounts for most of the return from the subdivision.

3. TOTAL CONSTRUCTION COST PER SINGLE FAMILY AREA ($/Ha) - This value can be compared for proposed subdivisions to determine which is less expensive considering single family area only. This is an important consideration since most subdivision return comes from the sale of single-family lots
4. TOTAL CONSTRUCTION COST PER PAVEMENT AREA ($/Ha) - This value, when compared for different proposed subdivisions, determines which is more efficient with respect to the total street area. This must be considered since pavement costs contribute significantly to the cost of constructing a subdivision.
5. TOTAL CONSTRUCTION COST PER STREET LENGTH ($/Ha) - When this value is used with the above value in comparing two proposed subdivisions, it gives an indication as to which proposal makes more efficient use of pavement. If the street widths for two proposed subdivisions are very different, this value unlike the value in the last section, will not consider this.
6. TOTAL CONSTRUCTION COST PER SALEABLE FRONTAGE ($/Ha) - This is one of the more important of the per unit construction cost values, since the saleable frontage is the portion of the subdivision which generates the most revenue by its sale.

7. TOTAL CONSTRUCTION COST PER TOTAL DWELLING UNIT ($/UNIT) - includes both single- and multi-family dwellings. This value, when compared for proposed subdivisions, indicates which is more cost-efficient with regards dwelling units.
8. TOTAL CONSTRUCTION COST PER SINGLE FAMILY UNIT ($/UNIT) - Like the single-family area unit cost above, this value is important because single-family units provide much of the return for a subdivision.

4.2.23 SUBDIVISION DENSITY

This section of the output data file describes the number of single- and multi-family dwellings per unit area. This data for this section is output in the following format:

FAMILY UNIT (Single, Multi or Total)	QUANTITY	AREA(Ha)	DENSITY(UNITS/Ha)

There are two parts to this section, with the above format applying to both. The first is based on gross area, while the second is based on net area. See the section above for a description of gross and net area.

4.2.24 YIELD EFFICIENCY

This section of the output data file describes the yield, which is a "frontage per unit" value. This data section includes the following:

1. SALEABLE FRONTAGE (m) - measured 6 m from the front property line on all single-family units, except for reverse pie-shaped lots which are measured 18 m from the front property line
2. STREET LENGTH (m) - total street length for the subdivision
3. GROSS SUBDIVISION AREA (Ha) - subdivision area neglecting commercial area.
4. YIELD:
    a) YIELD BASED ON TOTAL STREET LENGTH (m/m) - cost measurement which considers revenue in relation to street length
    b) YIELD BASED ON TOTAL SUBDIVISION AREA (m/Ha) - cost measurement which considers revenue in relation to subdivision area.

4.2.25 "INVESTMENT RATE OF RETURN VS. REVENUE" GRAPH

This section of the output data file is a graphical analysis of the investment rate of return. The graph plots the investment rate of return (%) on the vertical axis and revenue ($/m) on the horizontal axis.

This graph indicates the percent return on investment for a given revenue amount per meter of frontage.

4.2.26 MAINTENANCE COST

This section of the output data file summarizes the maintenance costs for the various subdivision elements. This section is output in the following format:

DESCRIPTION (Element Name)	QUANTITY	UNIT COST	MAINTENANCE COST

where UNIT COST is the unit maintenance cost. The above information is provided for the following subdivision elements:

1. WATERMAIN
2. DOMESTIC SEWER
3. STORM SEWER
4. PAVEMENT
5. SIDEWALK
6. BARRIER CURB AND GUTTER
7. ROLLED CURB AND GUTTER
8. BOULEVARD CURB AND GUTTER
9. PARK DEVELOPMENT
10. BUFFER DEVELOPMENT
11. CATCHBASIN
12. MANHOLE
13. STREET LIGHT
14. STREET SIGN
15. TRAFFIC SIGN

16. SINGLE-FAMILY DWELLING INFORMATION:

    a) GAS

    b) POWER

    c) TELEPHONE

17. MULTI-FAMILY DWELLING INFORMATION:

    (Same as for single-family dwellings.)

At the end of this data section, a total of maintenance costs for all the subdivision elements is provided.

4.2.27 CONSTRUCTION COSTS

This section of the output data file summarizes the construction costs for the various subdivision elements. The data in this section is output in the same format as the maintenance cost section above, except that the UNIT COST column is now construction unit costs and the MAINTENANCE COST column is replaced by the CONSTRUCTION COST column. A total for construction costs for the subdivision also appears at the end of this section.

4.2.28 OPERATION COSTS

This section of the output data file summarizes the operation costs for the various subdivision elements. The data in this section is output in the same format as the maintenance cost section above, except that the UNIT COST column is now operation unit costs and the MAINTENANCE COST column is replaced by the OPERATION COST column. A total for operation costs for the subdivision also appears at the end of this section.

4.2.29 DETAILED ANALYSIS

This section of the output data file summarizes present worth and the annual cost for various interest rates and subdivision life length. First, CONSTRUCTION COST, ANNUAL MAINTENANCE COST and ANNUAL OPERATION COST is output for the subdivision. The remainder of the output in this section has the following format:

INTEREST(%)	LIFE(YRS)	PRESENT WORTH($)	ANNUAL COST

4.2.30 CONSTRUCTION COST BREAKDOWN

This section of the output data file describes the cost breakdown for subdivision elements by account. The data for this section is presented in the following format:

DESCRIPTION (Element)	ACCOUNT NUMBER	PRESENT WORTH	ANNUAL COST

The above information is provided for all 21 subdivision elements which include:

1. WATERMAIN
2. DOMESTIC SEWER
3. STORM SEWER
4. PAVEMENT
5. SIDEWALK
6. BARRIER CURB AND GUTTER
7. ROLLED CURB AND GUTTER
8. BOULEVARD CURB AND GUTTER
9. PARK DEVELOPMENT
10 BUFFER DEVELOPMENT
11. CATCHBASIN
12. MANHOLE
13. STREET LIGHT
14. STREET SIGN
15. TRAFFIC SIGN

16. SINGLE-FAMILY DWELLING ELEMENTS:

a) GAS

b) POWER

c) TELEPHONE

17 MULTI-FAMILY DWELLING ELEMENTS:

(Same as single-family dwelling elements.)

Note that after the account data for each subdivision element are totals for both present worth and annual cost. At the end of this data section is also a total for PRESENT WORTH and ANNUAL COST for all subdivision elements.

4.2.31 MAINTENANCE COST BREAKDOWN

This section of the output data file describes the maintenance cost breakdown for the subdivision elements by account. The data output for this section appears in the same format as for construction costs above, but the PRESENT COST and ANNUAL COST columns now refer to maintenance costs. Totals for present and annual maintenance costs are output for each subdivision element and for the overall subdivision as for the construction cost breakdown.

4.2.32 OPERATION COST BREAKDOWN

This section of the output data file describes the operation cost breakdown for the subdivision elements by account. The data output for this section appears in the same format as for construction costs above, but the PRESENT COST and ANNUAL COST columns now refer to operation costs. Totals for present and annual operation costs are output for each subdivision element and for the overall subdivision as well as for the construction cost breakdown.

4.2.33 SUMMARY OF ACCOUNTS

This section of the output data file summarizes the construction, maintenance and operation costs by account. The data for this section is output in the following format:

DESCRIPTION (ACCN'T NAME)	ACCN'T #	CONST. COST	MAIN. COST	OP. COST

At the end of the summary are totals for construction, maintenance and operation costs for the complete subdivision.

CHAPTER 5

RACAM MODULES

5.1 FLOWCHART OF RACAM MODULES

```
RCMAIN
  |
  | <---------> HEAD
  |
  | <---------> CLEAR
  |
  | <---------> IOFILE
  |
  | <---------> INDAT1 <----------> IOFILE
  |
  | <---------> OUTDAT1
  |
  | <---------> CLEAR
  |
  | <---------> INDAT2 <----------> IOFILE
  |
  | <---------> OUTDAT2
  |
  | <---------> CLEAR
  |
  | <---------> ERRCHK <----------> PATHS1
  |
  | <---------> IDWATR <----------> PATHS2
  |                 |
  |                 |   <----------> PATHS3
  |
  | <---------> IDDMSW <----------> PATHS2
  |                 |
  |                 |   <----------> PATHS4
  |                 |
  |                 |   <----------> PATHS5
  |                 |
  |                 |   <----------> PATHS7
  |                 |
  |                 |   <----------> TAGPIT <-----> PATHS5
  |                 |
  |                 |   <----------> OPTPTH <-----> PATHS4
  |
  | <---------> IDSTSW
  |
  X
```

```
<---------> CREATE <----------> GROUP1 <-----> PATHS7
                |                   |
                |                   |   <-----> PATHS2
                |
                |   <----------> GROUP2 <-----> PATHS7
                                    |
                                    |   <-----> OPTPTH

<---------> ATTACH <----------> ADJ

<---------> WATER

<---------> DMSTC  <----------> DIAMTR
                |
                |   <----------> DEPTH
                |
                |   <----------> DSSDPTH

<---------> STORM  <----------> DIAMTR
                |
                |   <----------> DEPTH
                |
                |   <----------> SSSDPTH
                |
                |   <----------> ASPLIT

<---------> PAVEMENT <--------> ADJ
                |
                |     <--------> SAC
                |
                |     <--------> TUBE
                |
                |     <--------> COR
                |
                |     <--------> DMAKER <-----> CULCHK

<---------> MANHOLE <---------> CULCHK
                |
                |    <---------> DMAKER <-----> CULCHK
                |
                |    <---------> ASPLIT

<---------> LIGHT <-----------> COUNTER
                |
                |   <-----------> FINDPATH
                |
                |   <-----------> CHECKLEN
                |
                |   <-----------> CULCHK
                |
                |   <-----------> DMAKER <-----> CULCHK
                |
                |   <-----------> THREEWAY
                |
```

```
                |   <-----------> ASPLIT
<---------> CATBAS <----------> DRAIN
                |
                |   <----------> CATDEPTH
                |
                |   <----------> CATCALC
                |
                |   <----------> CATNODE <----> CULCHK
                |
                |   <----------> CORCHK
                |
                |   <----------> CATLIN <-----> CHECKLEN
                |                    |
                |                    |   <-----> CATDEPTH
                |                    |
                |                    |   <-----> CATCALC2
                |
                |   <----------> ASPLIT
                |
                |   <----------> CORNER <-----> CHECKLEN
                |                    |
                |                    |   <-----> CATNODE
                |                    |
                |                    |   <-----> CATLIN
<---------> SIDEWALK <--------> ASPLIT
<---------> SIGN <------------> DMAKER <-----> CULCHK
              |
              |   <------------> DMAKER2
              |
              |   <------------> ASPLIT
<---------> UTILITY
<---------> PARKBUFF <--------> ASPLIT
<---------> FRONTAGE
<---------> ANALYSIS <--------> SWTABLE
                 |
                 |    <--------> CLINE
                 |
                 |    <--------> ASPLIT
                 |
                 |    <--------> GRAPH
<---------> OMINIT
<---------> MAINCE <----------> GOGRAD
<---------> OPERAT <----------> GOGRAD
```

```
X
| <---------> ECONOM
|
| <---------> ACCOUNT <---------> ACCOUT
|
| <---------> ACCOUNT1 <--------> ACCOUT1
|
| <---------> ACCOUNT2 <--------> ACCOUT2
|
| <---------> ACCSUM
```

5.2 MAIN ROUTINE DESCRIPTION

5.2.1 RCMAIN - A Routine to Control the Execution of RACAM Procedures

The main program controls the output and the screen output or the interactive portion of the RACAM program. The subroutines HEAD and CLEAR control the screen. The subroutines INDAT1,INDAT2 read in the service element unit costs and the link and node data respectively. The routines OUTDAT1,OUTDAT2 output the unit cost and node and link data. The IOFILE routine creates the input and output files for the derived data that RACAM calculates. The HEAD routine creates and prints the RACAM heading in two inch letters on the screen.

The second function of RCMAIN is to set the following calibration data values used by the program:

* PEOPLE - the number of people per unit (multi- or single-family)
* DSVMIN - the minimum domestic sewer pipe velocity (m/s).
* SSVMIN - the minimum storm sewer pipe velocity (m/s).
* DSVMAX - the maximum domestic sewer pipe velocity (m/s).
* SSVMAX - the maximum storm sewer pipe velocity (m/s).
* WVMIN - the minimum water velocity in a pipe (m/s).

* FRICTN - 'Manning's' pipe roughness coefficient.
* DSCOVR - minimum depth of coverage for domestic sewer pipe (m).
* SSCOVR - minimum depth of coverage for storm sewer pipe (m).
* SPAMAN - maximum spacing between manholes (m).
* SPALIT - maximum spacing between lights (m).
* USE - domestic watermain use (m^3/person/day).
* FLOAD - water consumption safety factor.

5.3 DESCRIPTION OF SUBROUTINES CALLED BY THE MAIN ROUTINE

5.3.1 ACCOUNT - A Routine to Output Construction Cost Breakdown

The variables used by ACCOUNT are:

* TIME - number of times to overtype headings.
* WUNIT1 - the output device number code.
* MHCOST - manhole construction costs.
* LTCOST - light construction costs.
* INTRST - interest rate in percentage.
* ACCONT - the account name, associated with an element.
* ACCNUM - account number.
* DEFINE - full description of account.
* NINT - the number of different rates in INTRST.
* LIFE - the life of the construction element.
* WMCOST - water-main construction cost.
* DSCOST - domestic sewer construction cost.
* SSCOST - storm sewer construction cost.
* TCPAVE - pavement construction cost.
* WKCOST - sidewalk construction cost.
* BRCOST - barrier curb construction cost.
* RLCOST - rolled curb construction cost.
* PCOST - park development cost.
* BCOST - boulevard curb construction cost.
* CBCOST - catch-basin construction cost.
* MHCOST - manhole construction cost.

* LTCOST - street light construction cost.
* CSIGN - street sign construction cost.
* TRACOS - traffic light construction cost.
* GAS1 - gas line construction cost - single family unit.
* POW1 - power line construction cost - single family unit.
* PH1 - phone line construction cost - single family unit.
* GAS2 - gas line construction cost - multi-family unit.
* POW2 - power line construction cost - multi-family unit.
* PH2 - phone line construction cost - single family unit.
* TLCOST - total construction cost.
* AWORTH - annual worth.

ACCOUNT outputs the construction cost breakdown for each subdivision service element (watermain, domestic sewer, etc.) by account number. This routine outputs the requisite headings for the construction cost output, and outputs the interest rate and subdivision life. The routine ACCOUT which outputs the account numbers, and corresponding present and annual costs for each subdivision service element is then called. Total costs for each element and overall totals are provided by ACCOUT and ACCOUNT.

First, ACCOUNT outputs the heading for the construction cost breakdown. (Note: all output in this routine is written to WUNIT1 which is assigned to 6 in the main program and opened in a routine called IOFILE.) This heading is overtyped to highlight it, the number of overstrikes being determined by TIME. The interest rate, INTRST, and the subdivision life, LIFE, are then output.

Next, ACCOUNT outputs the headings for description, account number and costs, which are also overtyped. ACCOUNT alternately outputs the element description, then calls the routine ACCOUT which outputs the account name, ACCONT, the present cost, TYCOST, and the annual cost, TANNUL, by subdivision element. (See page 234, ACCOUT.) When calling ACCOUT, ACCOUNT passes the element name and the element cost variable. (eg: (1,WMCOST) is the argument passed to ACCOUT for watermain costs.)

For each subdivision element, ACCOUT outputs totals for present and annual construction costs, with this output being overtyped. As well, ACCOUNT outputs a net total for present and annual construction costs, TLCOST and AWORTH respectively, for all subdivision elements, with this output also being overtyped. Control is then returned to the calling program.

5.3.2 ACCOUNT1 - A Routine to Output Maintenance Cost Breakdown

The variables used by ACCOUNT1 are:

* TIME - number of times to overtype headings.
* MINST - interest rate for maintenance costs.
* MLIFE - useful life of the element.
* MNCOST - total elemental maintenance costs.
* MCAPAF - present value and annual present value of the maintenance costs.

ACCOUNT1 outputs the maintenance cost breakdown for each subdivision element (watermain, domestic sewer, etc.) by account number. This routine outputs the necessary headings for the maintenance cost output, as well as most of the maintenance cost data (the rest of the maintenance cost data is output by ACCOUNT2 and ACCOUT2).ACCOUNT1 then calls the routine ACCOUT1 which outputs the account numbers, and corresponding present and annual costs for each subdivision element. Total costs for each element are also provided by ACCOUNT1.

First, ACCOUNT1 outputs the heading for the maintenance cost breakdown. (Note: all output in this routine is written to WUNIT1 which is assigned to 6 in the main program and opened in a routine called IOFILE.) This heading is

overtyped to highlight it, the number of overstrikes being determined by TIME.

Next, ACCOUNT1 outputs the headings for description, account number and costs, which are also overtyped. ACCOUNT1 alternately outputs: the element description, the interest rate, MINST, and the service life, MLIFE, then calls the routine ACCOUT1 ,the present cost, MCAPAF(TYPE,J,1), and the annual cost, MCAPAF(TYPE,J,2), by subdivision element. (See page 186, ACCOUT1.) When calling ACCOUT1, ACCOUNT1 passes the element number. (eg: (1) is the argument passed to ACCOUT1 for watermain maintenance costs.)

For each subdivision element, ACCOUT1 outputs totals for present and annual maintenance costs, with this output being overtyped. Control is then returned to the calling program.

5.3.3 ACCOUNT2 - A Routine to Output Operations Cost Breakdown

The variables used by ACCOUNT2 are:

* TIME - number of times to overprint headings.
* OINST - interest rate for the operational costs.
* OPCOST - total elemental operational costs.
* OLIFE - useful life of the element.
* OCAPAF - present value and annual present value of the operational costs.

ACCOUNT2 outputs the operation cost breakdown for each subdivision element (watermain, domestic sewer, etc.) by account number. This routine outputs the necessary headings and data for most of the operation cost output, as well as some of the maintenance cost data(the rest of the operational cost data is output by ACCOUNT1 and ACCOUT1). It then calls the routine ACCOUT2 which outputs the account numbers, and corresponding present and annual operation costs for each subdivision element. Total costs for each element are also provided by ACCOUNT2.

First, ACCOUNT2 outputs the heading for the maintenance cost breakdown. (Note: all output in this routine is written to WUNIT1 which is assigned to 6 in the main program and opened in a routine called IOFILE.) This heading is

overtyped to highlight it, the number of overstrikes being determined by TIME.

Next, ACCOUNT2 outputs the headings for description, account number and operation costs, which are also overtyped. ACCOUNT2 alternately outputs the element description, the interest rate, OINST, and the service life, OLIFE, then calls the routine ACCOUT2 which outputs the account name, ACCONT, the present cost, OCAPAF(TYPE,J,1), and the annual cost, OCAPAF(TYPE,J,2), by subdivision element. (See page 186, ACCOUT1.) When calling ACCOUT2, ACCOUNT2 passes the element number. (eg: (1) is the argument passed to ACCOUT1 for watermain maintenance costs.)

For each subdivision element, ACCOUT2 outputs totals for present and annual maintenance costs, with this output being overtyped. As well, ACCOUNT2 outputs a net total of present and annual construction costs for all subdivision elements, with this output also being overtyped. Control is then returned to the calling program.

5.3.4 ACCSUM - A Routine to Output a Cost Summary by Account

Variables used by ACCSUM are:

* ACCOST - total construction costs by account.
* AMCOST - total maintenance costs by account.
* AOCOST - total operation costs by account.
* DEFINE - account description.
* ACCNUM - account number.
* TLCOST - total subdivision construction costs.
* TMCOST - total subdivision maintenance costs.
* TOCOST - total subdivision operation costs.

ACCSUM outputs total construction, operation and maintenance costs for each account. This routine also outputs net construction, operation and maintenance cost totals for the subdivision.

First, ACCSUM calculates the total subdivision cost for first account, TOTAL, by summing the construction costs, ACCOST , maintenance costs, AMCOST, and operation costs, AOCOST for it.

Secondly, ACCSUM outputs the headings for the cost summary. Each heading is overtyped to highlight it, with the number of overstrikes being determined by TIME.

Thirdly, the cost totals for the first account are output. The account description, DEFINE, the account number, ACCNUM, the construction cost total, ACCOST, the maintenance cost total, AMCOST, and the operation cost total, AOCOST, are all output for the first account.

Fourth, the value of TOTAL, as described previously, is calculated for the remainder of the accounts. In this same loop, the cost totals for the remainder of the accounts are output. The variables used for the account description and account name, are the same as those described for the first account. Control is then returned to the calling program.

5.3.5 ANALYSIS - A Routine to Output Data for All Subdivision Elements by Link

The variables used by ANALYSIS are:

* YSEC - number of rows in each section of the link description tables.
* COL - starting column for the link description tables.
* TIME - number of times heading are overtyped.
* DIASS - link diameter of storm sewer pipe (m).
* CSS - link unit cost of storm sewer pipe.
* SSCS - link storm sewer total cost (CSS x SSPIPE).
* SSPIPE - total subdivision storm sewer pipe length (m).
* SSCOST - total subdivision storm sewer pipe cost.
* LINE - used to define the table borders.
* DIADS - link diameter of domestic sewer pipe (m).
* CDS - link unit cost of domestic sewer pipe.
* DSCS - link domestic sewer total cost (CDS x DSPIPE).
* DSPIPE - total subdivision domestic sewer length (m).
* DSCOST - total subdivision domestic sewer cost.
* DIAWM - link diameter of watermain (m).
* CWM - link unit cost of watermain.
* WMCS - link watermain total cost (CWM x WMPIPE).

* WMPIPE - total subdivision watermain length (m).
* WMCOST - total subdivision watermain cost.
* XSEC - number of sections in the link description tables.
* XSPA - number of columns in each section of the link description tables.
* MCOL - maximum number of columns in the link description tables.
* LENGTH - length of the table section subheadings.
* IFRONT - starting column for the table section subheadings.
* LINE1-4 - used to define borders and titles for link description tables.
* TDSMAN - total subdivision domestic sewer manholes.
* TSSMAN - total subdivision storm sewer manholes.
* TSSCAT - total subdivision storm catchbasin.
* TDSCOS - total subdivision domestic sewer manhole cost.
* TSSCOS - total subdivision storm sewer manhole cost.
* TCBCOS - total subdivision catch basin cost.
* YTOTAL - flag to indicate when a table line division should be printed.
* DSMAN - number of domestic sewer manholes per link.
* DSCOSL - cost of domestic sewer manholes per link.
* SSMAN - number of storm sewer manholes per link.
* SSCOSL - cost of storm sewer manholes per link.

* LBASIN - number of storm sewer catch-basins per link.
* LCBCOS - cost of storm sewer catch-basins per link.
* NBASIN - number of storm sewer catch-basins per street length (#/m).
* CBCOST - total cost of storm sewer catch-basins
* SPACE - used to format subheadings for link description tables.
* TRAFIC - number of traffic signs per link.
* TRACOS - cost of traffic signs per link.
* LSTOP - number of stop signs per link.
* LYELD - number of yield signs per link.
* STCOST - cost of stop signs per link.
* YDCOST - cost of yield signs per link.
* LTLINK - number of street lights per link.
* LTCOSL - cost of street lights per link.
* NSIGN - number of street signs per link.
* SGCOST - cost of street signs per link.
* NLIGHT - number of street lights for subdivision.
* LTCOST - cost of street lights for subdivision.
* TSIGN - number of street signs for subdivision.
* CSIGN - cost of street signs for subdivision.
* NTRAFF - number of traffic signs for subdivision.
* TFCOST - cost of traffic signs for subdivision.
* CSCOST - cost of collector streets for subdivision.
* PVCOST - cost of all pavement for subdivision.
* RDPINT - road length per link (m).

* ALINK - road area per link (m^2).
* PVCOST - pavement cost per link.
* LSCOST - local street cost for subdivision.
* TPLEN - pavement length for subdivision (m).
* COLLEN - collector street length for subdivision (m).
* LOCLEN - local street length for subdivision (m).
* TPAREA - pavement area for subdivision (m^2).
* ACOL - collector street area for subdivision (m^2).
* TSRCOS - total sidewalk cost for subdivision.
* SRCOST - sidewalk cost for link.
* BARIER - barrier curb/gutter length per link (m).
* ROLLED - rolled curb/gutter length per link (m).
* BLVD - blvd. curb/gutter length per link (m).
* ASIDE - sidewalk area per link (m^2).
* TBR - barrier curb/gutter length for subdivision (m).
* TRL - rolled curb/gutter length for subdivision (m).
* TWK - sidewalk area for subdivision (m^2).
* TBL - blvd. curb/gutter length for subdivision (m).
* GAS1 - gas line construction cost - single family unit.
* POW1 - power line construction cost - single family unit.

* PH1 - phone line construction cost - single family unit.
* GAS2 - gas line construction cost - multi-family unit.
* POW2 - power line construction cost - multi-family unit.
* PH2 - phone line construction cost - single family unit.
* UTOTAL - number of family units per subdivision.
* GSSING - gas line cost for single family units per link.
* PWSING - power line cost for single family units per link.
* TSING - telephone line cost for single family units per link.
* GSMULT - gas line cost for multi family units per link.
* PWMULT - power line cost for multi family units per link.
* PHMULT - telephone line cost for multi family units per link.
* UTCOST - utility costs per link.
* VAL1-7 - total construction costs per link.
* WKCOSL - sidewalk cost per link.
* BRCOSL - barrier curb/gutter cost per link.
* BLCOSL - blvd. curb/gutter cost per link.
* RLCOSL - rolled curb/gutter cost per link.

* TCONST - total construction cost for subdivision.
* CNCOSL - total construction cost per link.
* XTOTAL - used to format link description table subheadings.
* FRONT - saleable frontage per link (m).
* MULTI - number of multi family units per link.
* UNITS - number of single family units per link.
* RAT1-4 - construction costs per: road area, street length, frontage, dwelling unit (multi and single).
* APARK - area of park development in subdivision (Ha.).
* PCOST - park construction cost per subdivision.
* ABUFF - area of buffer development in subdivision (Ha.).
* BCOST - buffer construction cost per subdivision.
* MNHOLE - number of manholes in subdivision.
* MHCOST - manhole cost for subdivision.
* TMULT - number of multi-family units per subdivision.
* TSING - number of single-family units per subdivision.
* TUSING - total utility cost for single-family units in subdivision.
* TUMULT - total utility cost for multi-family units in subdivision.

* TLCOST - grand total construction costs for subdivision.
* NCUL - number of cul-de-sacs per subdivision.
* NDEAD - number of dead ends per subdivision.
* N3WAY - number of 3-way intersections per subdivision.
* N4WAY - number of 4-way intersections per subdivision.
* NEXIT - number of subdivision exits.
* ASING - area of single family units per subdivision (Ha.).
* AMULT - area of multi family units per subdivision (Ha.).
* ASCHOL - area of schools per subdivision (Ha.).
* ACOM - area of commercial section per subdivision (Ha.).
* APAVE - area of pavement per subdivision (m^2).
* AWALK - area of sidewalk per subdivision (m^2).
* AOTHER - area of 'other' land per subdivision (Ha.).
* ASUB - area of total subdivision (Ha.).
* GROSS - gross area (Ha.).
* TLEN - street length (m).
* TCPTA - total construction cost per total area.
* TCPGA - total construction cost per gross area.
* TCPSA - total construction cost per single-family area.
* TCPPV - total construction cost per pavement area.

* TCPNA - total construction cost per net area.
* TCPST - total construction cost per street length ($/m).
* TCPFG - total construction cost per saleable frontage ($/m).
* TCPDW - construction cost for total dwelling unit.
* TCPSG - total construction cost per single-family dwelling.
* TFRONT - saleable frontage (m).
* SIGPGA - single-family dwellings for gross area.
* GSING - single family unit for gross area.
* MULPGA - multiple-family dwellings for gross area.
* TOTPGA - total dwellings for gross area.
* SIGPNA - single-family dwellings for net area.
* MULPNA - multiple-family dwellings for net area.
* TOTPNA - total dwelling for net area.
* YSTLEN - yield based on total street length.
* YGROSS - yield based on total gross area.
* TCOS - actual and assumed total construction cost per saleable frontage.
* ASSUME - assumed total construction cost per saleable frontage.
* REVMAX - maximum revenue.
* REVMIN - minimum revenue.

ANALYSIS calculates and outputs information describing the subdivision elements for each link. Physical, as well as

economic information is output for each link, with the information being laid out in tabular form.

First, ANALYSIS outputs the link description tables for storm sewer, domestic sewer and watermain information. This begins with outputting the appropriate headings which are overprinted to highlight them. ANALYSIS then calls SWTABLE which outputs physical and economic information, as well as formatting the table itself. (See page 335, SWTABLE.) The above procedure is repeated three times: for storm sewer, domestic sewer and watermain information. This results in the output of three separate tables.

Second, ANALYSIS does calculations and outputs the link description tables for manholes/catch-basins, street lights/street lengths/traffic signs, pavement, sidewalks/curbs, family units and link construction costs. For each of these tables, the number of sections, XSEC, and the number of spaces in each column, XSPA, are first defined. The routine, CLINE, is then called to define the variables (LINE1 to LINE4, XTOTAL) used in outputting the table borders. (See page 261, CLINE.) The maximum number of columns in the table, MCOL, is then calculated.

ANALYSIS does calculations allowing the section headings to be centered in each section, using the character string LINE3, and the variables LENGTH, and IFRONT. Subheadings are then assigned in a similar manner to the section headings,

but the subheading are inserted as a substring of LINE4, while the section headings are assigned as a substring of LINE3. The upper portion of each table is then output by WRITEing the character strings LINE, and LINE1 to LINE4.

Next, the totals for each section in the tables are initialized where necessary then calculated using loops. In these same loops, the subdivision element information for each link is output for the tables, with calculations of this link information being done where necessary. After outputting the information for each link, the totals for each section of the table are output. Note that there is a separate section of code for the output of each table.

Finally, several summaries are output: general quantity and cost, site-specific information, subdivision construction cost, subdivision density and yield efficiency. Each summary is basically an overview of the information presented in the link description tables.

The quantity and price summary section of ANALYSIS outputs quantity (length, or area, or number of units) and total costs for each of the subdivision elements. These values that are output have either been calculated previously in ANALYSIS, or in other routines.

Next, ANALYSIS outputs the site specific information summary. The number of each of the major road elements (NCUL, NDEAD, etc.) are output for the subdivision. Analysis

also outputs the land use distribution in terms of areas. (ASING, AMULT, etc.) These values, as well as those used indicating the number of road elements, have also been calculated previously in ANALYSIS or in other routines.

ANALYSIS then outputs a summary of subdivision construction costs. In order to create this summary, several cost per unit construction costs (eg. TCPPV, which is the total construction cost per hectare of pavement) are calculated. The total construction cost for the subdivision is printed out, followed by the cost per unit construction costs calculated previously.

ANALYSIS outputs the subdivision density summary next. In order to do so, the gross density (number of dwelling units per gross area) is calculated for single- and multi-family dwellings, as well as for all dwellings (SIGPGA, MULPGA and TOTPGA respectively). Net density (number of dwelling units per net area) is also calculated for single- and multi-family dwellings, and for all dwellings (SIGPNA, MULPNA and TOTPNA respectively). ANALYSIS then outputs the quantity of each of the dwelling type, TSING and TMULT, the area occupied by each dwelling type, GSING and GMULT for gross area and ASING + AMULT for net area, as well as the density is output for gross and net density. Totals for numbers of units, area and density are also output for both gross and net densities.

ANALYSIS now outputs a yield efficiency summary. Saleable frontage, TFRONT, total street length, TLEN and gross subdivision area, GROSS are first output for the summary. ANALYSIS then outputs the yield efficiency, including that based on total street length, YSTLEN, and that based on gross area, YGROSS. All of these values output have been calculated previously.

Finally, ANALYSIS calls a routine, GRAPH, which plots a graph of investment rate of return against revenue. (See page 293, GRAPH). After GRAPH returns control to ANALYSIS, control is returned to the calling program.

5.3.6 ATTACH - A Routine to Initialize Variables for the PAVEMENT Routine

The variables used by ATTACH are:

* ALINK - area of road on the link.
* AINTS - area of an intersection.
* CORWD - corresponding road width of a given link.
* LROAD - length of road on a given link.
* DSLINK - domestic sewer elevation of a given link.
* SSLINK - storm sewer elevation of a given link.
* EXIT - a flag indicating the exit node.
* L,N,J - loop counters.

This is a supplementary routine of PAVEMENT, it will initialize all the calculated variables to zero (LROAD, ALINK, CORWD, EXIT, AINTS) and the elevation variables to 9999.9 (SSLINK, DSLINK).

5.3.7 **CATBAS - A Routine to Calculate Number and Cost of Catchbasins**

The variables used by CATBAS are:

* BASHGT - height between the lead and the bedding (m).
* BASBED - height of the bedding (m).
* OFFSET - distance from center of street (m).
* SPACAT - spacing of catchbasins (m).
* CBASIN - number of catchbasins in subdivision.
* CBCOST - cost of catchbasins in subdivision.
* NNODE - number of nodes in subdivision.
* CBCOS - cost of catchbasins for the node.
* BASIN - number of catchbasins for the node.
* FLINK - flow link flag.
* STRCAT - number of catchbasins per street length (#/m).
* NLINK - number of links in subdivision.
* LBASIN - number of catchbasins per link.
* LCBCOS - cost of catchbasins per link.
* AVEDEP - average link depth (m).
* NTYPE - node type.
* NFLOW - number indicating the flow paths of the node to the catchbasins.
* NL - number of links associated with a node.
* CLINK - current link in memory.

* ATTLIN - links associated with the given node.
* WROAD - width of road of a given link (m).
* FLAG - indicates the direction of drainage flow.
* ERRFLG - flag to indicate that node has no drainage.
* WUNIT2 - device number for the error file.
* CBLEAD - length of catchbasin lead (m).
* CLEAD - minimum length of catchbasin lead (m).
* CBDPTH - indicates depth of catchbasins or non-existence of catchbasin (CBDPTH=0m).
* CDPTH - depth of catchbasin lead (m).
* BRANCH - one of a number of links attached to the node used to indicate the end of a PATH when it equals 1.
* NBL - number of links between 2 intersections.
* BULB - flag indicating bulb curve(s) presence.
* TLCOST - grand total construction costs for subdivision.

CATBAS is used to calculate the total number of catchbasins required for the subdivision. The total construction costs for the catchbasins in the subdivision are also calculated by this routine.

First, CATBAS initializes the variables. Next the number of catchbasins for each intersection are found by identifying: the intersections by checking the NTYPE of the nodes being processed, easements by checking the WROAD of the ATTLIN of the node being processed, and identifying

which links drain into the intersection or node by calling DRAIN,. It is assumed that each link attached to the node justifies a catchbasin. The routine checks for the existence of no drainage by calling the DRAIN routine. If there is no drainage then an error message is printed (eg. 'CATBAS-W-No drainage...).

Each ATTLIN are checked to see if it is an easement, if it is then CBLEAD equals 9999999.99 and CBDPTH is assigned the value of 0. If it is not then CATDEPTH is called to calculate the catchbasin lead lengths and CBLEAD equals CLEAD returned from CATDEPTH and CBDPTH equals CDPTH.

Second, the costs of the catchbasins are found by calling CATCALC to calculate catchbasin construction costs for a node. If the NTYPE of the node being processed is 1 then CATNODE is called to calculate the cost of catchbasins.

Third, the catchbasins required at the nodes are allocated to links. Then each intersection is processed in a do-loop. Each node prior to the node being processed N2, assigns BRANCH to equal 4, the last node through the loop is the node being processed N1, and it assigns the value of 1 to BRANCH. As long as STRLEN of both of these nodes is not 0 and BRANCH is greater than 0 then BRANCH is decremented by 1. NBL is initialized to 0 and then incremented at a numbered command. CLINK becomes the NBLth link of the PATH between 2 intersections, an intersection and a dead-end, and an intersection and a cul-de-sac. NBL is incremented until

CLINK of 0 is found at which point NBL is decremented and CORCHK is called to check the nodes to determine if they are bulb corners. If the BULB returned from CORCHK is 0 CATLIN is called to calculate the number of catchbasins per link. If BULB is not 0 CORNER is called to calculate the number of catchbasins on the bulb corner.

Fourth, the costs of the catchbasins for the link are summed. CBCOST is incremented by LCBCOS and CBASIN by LBASIN. These variables are calculated in the CATCALC routine. The costs are allocated to the appropriate accounts using the ASPLIT routine and TLCOST is incremented by CBCOST. The catchbasin amounts and costs are output. Control is then passed to the calling program.

5.3.8 CLEAR - A Routine to Clear the Terminal Screen and Print the RACAM Title

The variables used by CLEAR are:

* ESC - output escape code (used to indicate that special control characters will follow).
* BLA - control code used for screen manipulation.
* SH,Q - control codes used to home the cursor.
* J - control codes used to clear the screen.

CLEAR is used to clear the terminal screen, home the cursor, then print the RACAM heading. This routine is for VT-100 terminals.

After initializing the variables, CLEAR writes the a set of control codes. This set of control codes causes the terminal screen to clear. Second, CLEAR writes a second set of control codes. This set of control codes causes the cursor to home (go to the top left of the screen). CLEAR then prints the RACAM heading at the top of the screen. Control is then returned to the calling program.

5.3.9 CREATE - A Routine to Create Tributary Data

The variables used by CREATE are:

* TBTRYW - watermain tributary data returned from GROUP1.
* TBTRYD - domestic sewer system data returned from GROUP2.
* TBTRYS - storm sewer system data returned from GROUP2.
* NWMLNK - number of watermain links.
* WLINKS - links involved in the watermain system.
* NDSLNK - number of domestic sewer links.
* DLINKS - links involved in the domestic sewer system.
* NSSLNK - number of storm sewer links.
* SLINKS - links involved in the storm sewer system.
* CHONUM - number of choices that were made in designing a particular system.
* CHOLNK - link numbers of links involved in each choice.
  - (index from 1 to CHONUM, 1-3).
  - 1 is the number of the link not chosen.
  - 2 is the number of the link chosen.
  - 3 is the number of the node at which the choice occurred.

* ERRFLG - a flag to indicate whether error messages should be output to the screen (ERRFLG=0), to the error file (ERRFLG=1), or both (ERRFLG=2).
* TRIFLG - a flag to indicate whether the tributary data should be output or not (0=output, 1=no output).
* L - loop index.
* X,X1,X2 - miscellaneous variables.

CREATE creates tributary data for the watermain, domestic sewer, and storm sewer systems from the list of links involved in each system. Tributary data are data which organize the links into related groups of feeding and receiving links so that the flow in each system can be traced from origin to destination. The tributary data for each respective system consists of many groups, each composed of 2 to 4 links. There may be 1-3 'from' links (links supplying flow) in a group, but there must always be one 'to' link (link receiving flow) in a group.

First, CREATE organizes the link descriptors of each link so that the node of higher elevation is in the array NODE1 and the node of lower elevation is in the array NODE2. This node arrangement corresponds to the data format requirements of succeeding subroutines.

Second, CREATE calls GROUP1 to determine the tributary data for the watermain system; the results of GROUP1 are stored in TBTRYW.

Third, CREATE calls GROUP2 to determine the tributary data for the domestic sewer system. These results are stored in TBTRYD. GROUP2 also returns the path choices that were made while processing the data including the number of the link not chosen, the number of the link chosen, and the node at which the choice occurred. CREATE outputs this data to the error file as specified by ERRFLG (i.e. to the screen, error file, or both). Note: The path choices made by GROUP2 cannot be changed at present. The user will have this ability in the future.

Fourth, CREATE calls GROUP2 to determine the tributary data for the storm sewer system, the results are stored in TBTRYS. Again, CREATE outputs the path choice information, as described above.

Finally, CREATE outputs the tributary data to the data file. Output is controlled by TRIFLG: 0=output; 1=no output. Control then returns to the calling routine.

5.3.10 DMSTC - a Routine to Design the Sewer System and Find the Associated Costs.

The variables used by DMSTC are:

* DSCOVR - minimum depth of coverage for sewer pipe (m).
* DSDPTH - depth of sewer (m).
* DSCOST - total cost of sewer.
* QPCDS - sewer flow per capita. (455 liters per capita converted to cms per capita as obtained from Hamilton/Wentworth.)
* SLOPE - slope between node1 and node2.
* DSLOPE - slope of sewer pipe between origin and destination node.
* DSELEV - elevation of sewer pipe (m).
* TDAREA - total drainage area of each destination link (Ha.).
* A1 - total drainage area of each link (acres).
* X1 - total drainage area of each link (Ha.).
* QDS - total generated sewer flow (cfs).
* D - calculated diameter of pipe (m).
* D2 - required diameter of sewer pipe (m).
* DIADS - diameter of sewer pipe (m).
* INDX2 - diameter index for sewer(1-5).
* DINDX2 - depth index for sewer (1-9).
* CDS - sewer unit link cost.

* TBTRYD - locations (1-3) contain the origin links describing the domestic sewer tributaries. Location 4 contains the destination link. Temporarily assigned to the variable CONST.
* NDSGRP - number of tributary groups in domestic sewer system.
* TPOP - total population feeding each destination link.
* PDS - initial population of link for domestic sewer calculation.
* P1 - population of a tributary link.
* UNITS - number of single family units per link.
* MULTI - number of multi family units per link.
* FM - empirical formula result.
* E1-2 - elevation of origin and destination nodes of TBTRYD link.
* LLINK - link length (m).
* DES1-2 - ground elevation of the sewer pipe at the node (m).
* ELEV - ground elevation of the node (m).
* VEL - actual velocity of water in pipe (m/s).
* DEPTH1-2 - elevation of the origin and destination nodes minus the elevation of the pipes (m).
* ADPTH - average depth into ground of pipe (m).

The DMSTC routine calculates the sewer system parameters and costs.

First, for each tributary group DMSTC checks the origin links, if they are less than or equal to zero, then PDS(TBTRYD) of the destination link is incremented by TPOP and the next NDSGRP is processed. If the origin links are greater than zero TOLINK is set to zero. If the the origin links equal the destination link then TOLINK equals one. P1 is calculated and added to TPOP.

P1 = [UNITS(TBTRYD)+MULTI(TBTRYD)]*PEOPLE +PDS(TBTRYD)

Secondly, DMSTC calculates the generated domestic sewer flow. FM and QDS are calculated:

$FM = 5.0 / [(P1/1000)^2]$

QDS = P1 * QPCDS * FM.

FM is checked and values greater than 5 are recued to 5, values less than 2 are increased to 2. Next GSLOPE is calculated:

GSLOPE = (E1 - E2) / LLINK(TBTRYD)

GSLOPE is checked to see if its value is greater than zero, if so then SLOPE IS EQUAL to 0.000169, if not SLOPE equals GSLOPE and DSLOPE equals SLOPE. DD and D are calculated:

$DD = D = [3.208*QDS*FRICTN/SQRT(DSLOPE)]^{0.375}$

Thirdly DD is checked and INDX2 is assigned. If DD is less than 0.200, 0.250, 0.300 D2 is sequentially assigned to 0.200, 0.250, 0.300 and INDX2 to 1, 2, 3. Otherwise D2

equals 0.375 and INDX2 equals 4. DIADS of the tributary link ,TBTRYD, is equal to D2.

Fourth DMSTC calls DIAMTR to determine the pipe slope required to meet or exceed the minimum acceptable water velocity in a pipe. If VEL is greater than DVSMAX, and D2 is less than 0.375 then the diameter of the pipe is not appropriate for the water speed within it and it both must be changed. DSLOPE equals SLOPE, and DD equals D2 and DD is checked and INDX2 is reassigned. If DD is less than 0.200, 0.250, 0.300 D2 is sequentially assigned to 0.200, 0.250, 0.300 and INDX2 to 1, 2, 3. Otherwise D2 equals 0.375 and INDX2 equals 4. This loop continues until the appropriate velocity and diameter is calculated. DSDPTH and DSE1 are calculated:

DSDPTH = DSCOVR + DIADS(TBTRYD) + 0.100

DSE1 = ELEV[NODE1(TBTRYD)] - DSDPTH

and TOLINK is checked, if it is greater than 0.5 it checks DSELEV and continues as explained in the next paragraph. If TOLINK is less than 0.5, then DSE2 equals:

DSE2 = DSE1 - LLINK(CONST) * DSLOPE

If DSELEV[NODE2(CONST)] is greater than DSE2, then DSELEV[NODE2(CONST)] equals DSE2. DEPTH is called to determine the depth index of a given pipe. If DPTH2 * 1.001 is greater than DSDPTH then:

DSE1 = DSE1 - DSDPTH + DPTH2

and the DMSTC loops back to recalculating DSE2 with a new value of DSE1. If DPTH2 * 1.001 is less than DSDPTH then DMSTC continues executing.

Fifth the DSELEV of the origin node of the tributary link is checked against DSE1. If it is greater than DSE1 than it is assigned the value of DSE1. DSE2 is calculated as being:

DSE2 = DSELEV[NODE1(TBTRYD)] - LLINK(TBTRYD) * DSLOPE

If DSELEV[NODE2(TBTRYD)] is greater than DES2 then DES2 is assigned the value of DSELEV. DEPTH is called to determine the depth index of a given pipe. If DPTH2 * 1.001 is less than DSDPTH then:

DSELEV[NODE1(TBTRYD)]=DSELEV[NODE1(TBTRYD)]-DSDPTH+DPTH2

and DMSTC loops back to the beginning of the fifth section, otherwise it continues to the sixth section which calculates costs.

Sixth, the costs of the excavation are calculated and stored or passed to the appropriate accounts by calling ASPLIT. CDS is equal to the COST12, which is dependent upon the depth indexes. DSCOST is incremented by the value of CDS*LLINK of the tributary link being processed and DIADS is multiplied by 1000. The following values are output:

CONST, P1, QDS, GSLOPE, D, VEL, DSLOPE, DPTH1, DPTH2, ADPTH

DSSDPTH is called to calculate the excavation depth for the pipe and PDS is calculated as explained in the first section. Control is then returned to the calling program.

5.3.11 ECONOM - A Routine to Calculate the Present Value Totals for Operational and Maintenance Costs

The variables used by ECONOM are:

* TLCOST - total subdivision construction costs.
* TMCOST - total subdivision maintenance costs.
* TOCOST - total subdivision operation costs.
* OPCOST - present, future, annuity values returned from GOGRAD, stored in an array.
* MNCOST - total elemental maintenance costs.
* TACOST - total annual operation and maintenance cost.
* NINT - total number of interest rate values.
* PW - the present value of a particular cost returned from the UNFORM1 function.
* AC - the present value of an annual cost returned from the UNFORM function.
* INTRST - interest rate in percentage.
* LIFE - the life of the construction element.
* RI - the interest rate in decimal terms.

The ECONOM routine calculates the totals for the operational and maintenance costs for the subdivision. The routine adds the operational, TOCOST, and maintenance, TMCOST, costs then totals them in TACOST.

The present and annual costs are calculated by using the UNFORM functions (See page 348, for an explanation of the function.).

ECONOM writes all of these results to the output file.

5.3.12 ERRCHK - A Routine to Check for Data Errors

The variables used by ERRCHK:

* NNODE - the number of nodes in a subdivision.
* NTYPE - The type of node: cul-de-sac, dead-end, 3-way, etc. If a node is an access/egress node, then NTYPE will initially have a negative value; later, when the access/egress points no longer need to be identified, the negative values are converted to positive values.
* NLINK - the number of links in the subdivision.
* NODE1 - an array which holds the node of highest elevation for each link.
* NODE2 - an array which holds the node of lowest elevation for each link.
* ERRFLG - a flag to indicate whether error messages should be output to the screen (ERRFLG=0), to the error file (ERRFLG=1), or both (ERRFLG=2).
* CHEKED - an array which keeps track of which links are directly or indirectly attached to an access/egress point (i.e. continuous with an exit). If a link is continuous, its corresponding element in CHEKED is flagged with a value of one.

* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.
* BRANUM - the number of branches (forks in the path) encountered during a given continuity test of a link.
* BRALNK - an array holding the link number of the link which occurred before each branch.
* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the link numbers of all the possible paths.
* FLAG1 - if set to one, signifies an end to the continuity test of a given link.
* FLAG2 - if set to one, signifies that REFLNK should return to the most recently encountered branch and try another path.
* L,L1,L2,I - loop index variables.
* X,X1,X2 - miscellaneous variables.

ERRCHK checks the validity of the input data and outputs the appropriate error message when an error is detected.

First, ERRCHK checks for out of range link descriptor values, that is, node values which are greater than the number of nodes in the system. A simple loop steps through the link descriptors of each link and compares these descriptors to the total number of nodes in the system (NNODE). When a node descriptor is greater than NNODE, then the following error message is output (in a manner defined by ERRFLG): ERRCHK-I-Link #xx references an out of range node.

Second, ERRCHK checks for links which have the same link descriptor at each end. A simple loop steps through the links and compares the descriptors of each link. If the descriptors are the same, the following error message is output (in a manner defined by ERRFLG): ERRCHK-I-Link #xx has the same node at each end.

Third, ERRCHK checks for links which have the same pair of link descriptors (end points). For example, if one link has the end points 4 and 7, and another link has the end points 7 and 4, or 4 and 7, an error would be detected. This check is performed by two nested loops. The first loop steps through the links, and the second loop compares each link to the one indexed in the first loop. If two links having the same link descriptors are detected, the following error

message is output (in a manner defined by ERRFLG): ERRCHK-I-Link #xx has the same nodes as link #yy. Note that if an error of this type occurs, two error messages will be displayed, one message for each link in the offending pair.

The fourth and most important task performed by ERRCHK is to verify that every link is directly or indirectly attached to an access/egress point (i.e. continuous with an exit). A loop is used to index each link for testing. If a link was tested in connection with the test of a prior link, (CHEKED indicates whether the link has already been tested) then the test for this link is not repeated and the loop continues to the next link. To determine whether a link is directly or indirectly attached to an access/egress point, REFLNK is initially set to the link indexed by the loop. Then PATHS1 is called to identify all possible paths emerging from the current link (PATHS1 only returns those paths which have not already been tested). If more than one path choice exists, the link number to which REFLNK is pointing is stored in BRALNK, a path is selected and the test repeated with REFLNK pointing to the newly chosen link. As REFLNK is moved from link to link, each link it points to is flagged in CHEKIN. If a dead end (i.e. no path choices) is reached, REFLNK is reset to the most recent link where a branch was identified. When an access/egress point has been found, the original link and every link accessed by REFLNK (as recorded in CHEKIN) is flagged in CHEKED, all CHEKIN flags are cleared, and the next link is indexed for testing.

If all branches have been tried and no paths leading to an access/egress point can be found, the following error message is output: ERRCHK-I-Link #xx has no path to an exit. Note that when one of these errors occurs, several other errors may occur as well. These errors may often be corrected by: (1) correcting a previously identified error, (2) ensuring that no links are arranged in a loop without being joined to the rest of the system, and (3) ensuring that all access/egress points are properly identified (i.e. node type values have negative signs). Also, note that if none of the access/egress paths were properly identified then all links will produce this error message.

Finally ERRCHK organizes the link descriptors of each link so that the node of higher elevation is in array NODE1 and the node of lower elevation is in array NODE2. This node arrangement corresponds to the data format requirements of succeeding subroutines. Control then returns to the calling routine.

5.3.13 FRONTAGE - A Routine to Calculate Saleable Frontage Area and Yield Efficiencies for the Subdivision

The variables used by FRONTAGE are:

* FRATIO - a constant ratio used to calculate the saleable frontage for school and multi-family units.
* CONS - 10000.00 constant divisional factor.
* APAVE - pavement area for subdivision (m^2).
* TAREA - total area of the pavement (m^2).
* AWALK - sidewalk area for subdivision (m^2).
* TWK - total sidewalk length in the subdivision (m).
* TFRONT - total saleable frontage for the subdivision (m).
* ASING - total area of single family units (m^2).
* ASUB - total area of the subdivision (m^2).
* APARK - total area of the parks (m^2).
* ABUFF - total area of the buffers (m^2).
* ASCHOL - total area of the schools (m^2).
* AMULT - total area of multi-family units (m^2).
* ACOM - total area of the common areas in subdivision (m^2).
* AOTHER - total area of 'other' areas in subdivision (m^2).
* FRONT - saleable frontage of a given link (m^2).
* NLINK - number of links in the subdivision.
* GROSS - the gross subdivision area (m^2).

* YSTLEN - yield efficiency based on total street length (TFRONT/TLEN).
* TLEN - total street length in the subdivision (m).
* YGROSS - yield efficiency based on subdivision's gross area (TFRONT/GROSS) (m/m^2).

FRONTAGE determines the total saleable frontage available for the subdivision. It also calculates yield efficiency factors based on the street length and gross area in the subdivision.

First the routine calculates the single family area in the subdivision by subtracting all other areas from the total area.

Second the total frontage lengths (TFRONT) are calculated by summation of:

TFRONT = FRONT + ASCHOL * FRATIO + AMULT * FRATIO

The gross area of the subdivision is :

GROSS = ASUB - ACOM - AOTHER

5.3.14 **HEAD- A Routine to Clear the Terminal Screen and Print the RACAM Title.**

The variables used by HEAD are contained only in HEAD and are not passed or used in any other routine. Various constants are defined in the routine to define the screen size.

First, the cursor is 'homed' to bring it to a particular position on the screen. Second, the screen is cleared. Third, the RACAM heading is printed using a series of formatted asterisks to print out the RACAM title.

5.3.15 IDDMSW - A Routine to Identify Links Comprising the Domestic Sewer System

The variables used by IDDMSW are:

* NTYPE - The type of node: cul-de-sac, dead-end, 3-way, etc. If a node is an access/egress node, then NTYPE will initially have a negative value; later, when the access/egress points no longer need to be identified, the negative values are converted to positive values.
* NODE1 - an array which holds the node of highest elevation for each link.
* NODE2 - an array which holds the node of lowest elevation for each link.
* NEWELV - a copy of ELEV that can be modified.
* OLDELV - a copy of NEWELV to reverse modifications.
* CHEKED - an array which keeps track of which links directly or indirectly attach to access/egress points (i.e. continuous with an exit). If a link is continuous, its corresponding element in CHEKED is flagged with a value of one.
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* PCHIDX - index into POSCHN.

* POSCHN - an array containing the links possibly belonging to the chain currently being developed.
* CHNIDX - index into CHAINS.
* CHAINS - a two-dimensional array containing a list of chains and the links which make up each chain. (A chain is a connected group of links.) Any chain that leads to an access/egress point is identified with a one in the zeroth element of the array.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.
* BRANUM - the number of branches (forks in the path) encountered during a given continuity test of a link.
* BRALNK - an array holding the link number of the link which occurred before each branch.
* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the link numbers of all possible paths.

* FLAG1 - indicates how many times the pit routine has been done. The routine will be forced to end if this reaches 1000.
* FLAG2 - if set to one, signifies an end to the continuity test of a given link.
* FLAG3 - if set to one, signifies that REFLNK should return to the most recently encountered branch and try another path.
* ERRFLG - a flag to indicate whether error messages should be output to the screen (ERRFLG=0), to the error file (ERRFLG=1), or both (ERRFLG=2).
* RESULT - indicates whether the current path choice has been tried before. If RESULT=0, the current path choice has not been tried. If RESULT=1, the current path choice has been tried; this path would result in the duplication of an existing chains.
* LNKTYP - if a link is directly or indirectly connected to an access/egress point via links which are obviously part of the domestic sewer system LNKTYP (link type) is set to zero. If a link is not continuous, LNKTYP is set to 1.
* CHECK - an array used to keep track of links which are obviously in the domestic sewer system.

* LENGTH - the sum of the lengths of each individual link in a chain.
* NDSLNK - number of domestic sewer links.
* DLINKS - an array holding the links involved in the domestic sewer system.
* L,L1-3,I - loop index variables.
* X,X1,X2 - miscellaneous variables.

IDDMSW identifies links comprising the domestic sewer system. Since the domestic sewer and storm sewer systems operate primarily under gravity flow conditions, IDDMSW must eliminate all pits from the sewer system. (A pit is a node with inflows but no outflows; that is, the elevation of a node is lower than the elevations of all adjacent nodes.) First, the trenching elevation of each node, obtained by subtracting the trenching depth (i.e. 2.75 m) from the natural ground elevation (ELEV), is stored in NEWELV and OLDELV.

Then a DO-WHILE loop is set up for the pit elimination procedure. This loop will continue to execute as long as FLAG1 remains less than 1000. Within the DO-WHILE loop, another loop indexes nodes in the subdivision. REFNOD is set to the point to the indexed node and TAGPIT is called. If TAGPIT indicates that REFNOD is not a pit (i.e. PITFLG=0), then the next node is indexed. However, if REFNOD is a pit

(i.e. PITFLG=1), then an attempt is made to eliminate the pit.

PATHS5 is called to identify nodes adjacent to the pit. These nodes are systematically lowered by 0.02 m. Their new elevations are compared to REFNOD elevation with a call to TAGPIT. If one of the adjacent nodes is not a pit then the number of this node is stored in X1. The elevations of the remaining adjacent nodes are restored to their original values. If none of the adjacent nodes can be lowered by 0.02 m to eliminate the pit, then REFNOD is raised 0.02 m above the lowest adjacent node. Raising REFNOD may result in the formation of a pit elsewhere; however, these new pits will probably be eliminated during the 1000 iterations of the pit elimination section of IDDMSW. When all pits have been eliminated or FLAG1=1000, ELEV is updated with the new elevations.

Then the domestic sewer links are identified. First, link descriptors of each link are organized so that the node of higher elevation is in array NODE1 and the node of lower elevation is in array NODE2. Node arrangement corresponds to the data format requirements of succeeding subroutines.

Second, the links which are obviously part of the domestic sewer system are identified. A loop indexes each link in the subdivision so that the presence of single and multi family units and initial conditions may be detected. When family units and initial conditions are present on the

indexed link, the link number is copied into DLINKS and its corresponding element is flagged in CHECK.

Third, each link in the domestic sewer system is tested to determine whether it is directly or indirectly connected to an access/egress point via other domestic sewer links. However, a new criteria is introduced; the path must be downhill. A loop is used to index each link for testing. If a link was tested in connection with the test of a prior link (CHEKED indicates whether the link has already been tested), then the test for this link is not repeated and the loop continues to the next link. To determine whether a link is directly or indirectly attached to an access/egress point, REFLNK is initially set to the link indexed by the loop. Then PATHS2 is called to identify all possible paths emerging from the current link (PATHS2 only returns those paths which have not already been tested). A loop is used to delete all upward sloping paths from the list of possible paths so that only downsloping paths remain. If more than one path choice exists, the link number to which REFLNK is pointing is stored in BRALNK, a path is chosen, and the test repeated with REFLNK pointing to the newly chosen link. As REFLNK is moved from link to link, each link it points to is flagged in CHEKIN. If a dead end is reached (i.e. no path choices), REFLNK is reset to the most recent link where a branch was identified. When all branches have been tried and no paths leading to an access/egress point can be found, LNKTYP is set to one indicating that a path must be created.

When an access/egress point is found, the original link and every link accessed by REFLNK (as recorded in CHEKIN) is flagged in CHEKED, all CHEKIN flags are cleared, and LNKTYP remains set to zero.

At this point, if LNKTYP=0, the loop finishes and the routine returns to test the next link. If LNKTYP=1, the routine identifies the shortest path from the given link to an access/egress point, adding all links along that path to the domestic sewer system. REFLNK is set to the current link. PATHS4 is called to identify valid domestic sewer links (including easement). Each link REFLNK points to is stored in POSCHN; links are stored in the order they were pointed to. When a dead end or access/egress point is encountered, or when REFLNK has pointed to 50 links, the contents of POSCHN are copied into CHAINS. (A one is stored in the zeroth element of CHAINS if an access/egress point is encountered.) The elements are then erased up to the last branch, a new chain is begun, and the procedure repeats. When all branches have been taken, (i.e. all chains have been found) the shortest chain capable of reaching an access/egress point is identified.

A loop is used to step through the chains so that chains ending in an access/egress point (i.e. zeroth element equals one) may be found. When such a chain is found, a second loop steps through the links in the chain and sums their lengths. The total length is stored in CHAINL in the

element corresponding to the chain number. When the length of all chains ending in an access/egress point have been calculated, a loop is used to isolate the shortest chain. The procedure assumes that the first chain possesses the shortest length, and stores the chain's number in X2. The length of each remaining chain is then compared to the length of chain X2. If a chain with a shorter length than chain X2 is discovered, the number of the shorter chain is stored in X2. When the loop ends, X2 identifies the shortest chain. Another loop flags the elements in CHECK corresponding to the links in the chosen chain.

Links that are used to find the shortest path but which were not previously included in the domestic sewer system are added to the system now. Note, an important difference between IDWATR and IDDMSW is that IDWATR does not update WLINKS until a path leading to an access/egress point has been identified for all links in the subdivision system. IDDMSW updates DLINKS after a path leading to an access/egress point has been identified for each link. The result is a priority given to existing links in the system when testing for continuity with an exit. Consequently, the domestic sewer routine does not include new links in the system if a path to an access/egress point along previously identified links already exists.

The routine returns to the beginning of the loop to test the next link. When all links have been tested, the

routine eliminates level links (i.e. links having the same elevation on both ends). These links must be eliminated so that the direction of sewer flow can easily be defined. A loop indexes links. PATHS7, is called to identify links adjacent to the nodes belonging to the indexed link. OPTPTH identifies the shortest path leading to an access/egress point. The link leading to the shortest path is connected to one of the corresponding nodes belonging to the level link. Then one of the nodes is lowered by 0.01 m. TAGPIT determines whether lowering this node creates a pit, in which case the opposite node is raised.

When a level link is eliminated, the flow situation may change. The routine consequently returns to the beginning and recreates the domestic sewer system using the modified elevations.

When all pits and level links are eliminated, the domestic sewer system has been defined and, control returns to the calling program.

5.3.16 IDSTSW - A Routine to Identify Links Involved in the Storm Sewer System

The variables used by IDSTSW are:

* NLINK - number of links in the subdivision.
* NSSLNK - number of storm sewer links.
* SLINKS - links involved in the storm sewer system.
* L - loop index.

IDSTSW identifies links involved in the storm sewer system. Every link in the subdivision is assumed to be part of the storm sewer system. Therefore, a loop is used to copy the link identifier into SLINKS. NSSLNK is initialized to NLINK.

5.3.17 IDWATR - A Routine to Identify Links Comprising the Watermain System

The variables used by IDWATR are:

* NTYPE - The type of node: cul-de-sac, dead-end, 3-way, etc. If a node is an access/egress node, then NTYPE will initially have a negative value; later, when the access/egress points no longer need to be identified, the negative values are converted to positive values.
* NODE1 - an array which holds the node of highest elevation for each link.
* NODE2 - an array which holds the node of lowest elevation for each link.
* CHEKED - an array which keeps track of which links are directly or indirectly attached to an access/egress point (i.e. continuous with an exit). If a link is continuous, its corresponding element in CHEKED is flagged with a value of one.
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* PCHIDX - index into POSCHN.

* POSCHN - an array containing the links possibly belonging to the chain currently being developed.
* CHNIDX - index into CHAINS.
* CHAINS - a two-dimensional array containing a list of chains and the links which make up each chain. (A chain is a connected group of links.) Any chain that leads to an access/egress point is identified with a one in the zeroth element of the array.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being test, and then moves along all possible paths in an attempt to reach an access/egress point.
* BRANUM - the number of branches (forks in the path) encountered during a given continuity test of a link.
* BRALNK - an array holding the link number of the link which occurred before each branch.
* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the link numbers of all possible paths.

* FLAG1 - if set to one, signifies an end to the continuity test of a given link.
* FLAG2 - if set to one, signifies that REFLNK should return to the most recently encountered branch and try another path.
* ERRFLG - a flag to indicate whether error messages should be output to the screen (ERRFLG=0), to the error file (ERRFLG=1), or both (ERRFLG=2).
* LNKTYP - if a link is directly or indirectly connected to an access/egress point via links which are obviously part of the watermain system, LNKTYP (link type) is set to zero. If a link is not continuous, LNKTYP is set to 1.
* CHECK - an array used to keep track of links which are obviously in the watermain system.
* LENGTH - the sum of the lengths of each individual link in a chain.
* NWMLNK - number of watermain links.
* WLINKS - an array holding the links involved in the watermain system.
* L,L1-3,I - loop index variables.
* X,X1,X2 - miscellaneous variables.

IDWATR identifies links comprising the watermain system. First, the links which are obviously part of the watermain system are identified. A loop indexes each link in the subdivision so that the presence of single- and multi-family units and initial conditions may be detected. When family units and initial conditions are present on the indexed link, the link number is copied into WLINKS and its corresponding element is flagged in CHECK.

Second, each link in the watermain system is tested to determine whether it is directly or indirectly connected to an access/egress point via other watermain links. A loop is used to index each link for testing. If a link was tested in connection with the test of a prior link, (CHEKED indicates whether the link has already been test), the test for this link is not repeated and the loop continues to the next link. To determine whether a link is directly or indirectly attached to an access/egress point, REFLNK is initially set to the link indexed by the loop. Then PATHS2 is called to identify all possible paths emerging from the current link (PATHS2 only returns those paths which have not already been tested). If more than one path choice exists, the link number to which REFLNK is pointing is stored in BRALNK, a path is selected, and the test repeated with REFLNK pointing to the newly chosen link. As REFLNK is moved from link to link, each link it points to is flagged in CHEKIN. If a dead end (i.e. no path choices) is reached, REFLNK is reset to the most recent link where a branch was identified. When all

branches have been tried and no paths leading to an access/egress point can be found, LNKTYP is set to one indicating that a path must be created. When an access/egress point is found, the original link and every link accessed by REFLNK (as recorded in CHEKIN) is flagged in CHEKED, all CHEKIN flags are cleared, and LNKTYP remains set to zero.

At this point, if LNKTYP=0, the loop finishes and the routine returns to test the next link.Note, however, that LNKTYP is immediately forced to one to prevent a watermain link on one side of the subdivision from being connected to an access/egress point on the opposite side. This situation is undesirable because the pressure available at an access/egress point on one side of the subdivision may not be large enough to adequately feed a link on another side. By forcing LNKTYP to be one, the shortest path to an access/egress point is found for every link, effectively rendering the continuity check useless. However, the continuity check remains a part of the routine since some situations may require that a minimum number of links be used without regard for pressure losses.

Therefore, since LNKTYP=1, the routine identifies the shortest path from the given link to an access/egress point, adding all links along that path to the watermain system.

First, a simple loop indexes all watermain links so that they may be tested for cul-de-sacs and dead-ends (i.e.

NTYPE=1). These links are then flagged in CHEKIN. PATHS3 is called to identify valid (non-easement and unused) watermain links adjacent to REFLNK. REFLNK is set to the current link. As REFLNK is moved from link to link, the number of each link it points to is stored in POSCHN; links are stored in the order they were pointed to. When a dead end, or access/egress point is encountered, or when REFLNK has pointed to 50 links, the contents of POSCHN are copied into CHAINS (a one is stored in the zeroth element of CHAINS if an access/egress point is encountered). The elements in POSCHN are then erased up to the last branch, a new chain is begun and the procedure repeats. When all branches have been taken, (i.e. all chains have been found), the shortest chain capable of reaching an access/egress point is identified.

A loop is used to step through the chain so that chains ending in an access/egress point (i.e. zeroth element equals one) may be found. Whenever such a chain is found, a second loop steps through the links in the chain and sums their lengths. The total length is stored in CHAINL in the element corresponding to the chain number. When the length of all chains ending in an access/egress point have been calculated, a loop is used to isolate the shortest chain. The procedure assumes that the first chain possesses the shortest length, and stores the chain's number in X2. The length of each remaining chain is then compared to the length of chain X2. If a chain with a shorter length than chain X2 is discovered, the number of the shorter chain is

stored in X2. Another loop flags the elements in CHECK corresponding to the links in the chosen chain.

Finally, the routine returns to the beginning of the loop to test the next link. When all links have been tested, the routine returns control to the calling program. It is important to note that except for easement links, the chains can be composed of any link type.

5.3.18 INDAT1- A Routine to Read the Unit Cost Data

The variables used by INDAT1 are:

* IST - flag indicating the naming of the input or output file.
* RUNIT1 - unit device number as dictated by RCMAIN.
* COST1-33 - the unit costs of the service elements (See page 208 OUTDAT1).
* PEOPLE - the number of people per unit (multi- or single-family)
* DSVMIN - the minimum domestic sewer pipe velocity (m/s).
* SSVMIN - the minimum storm sewer pipe velocity (m/s).
* DSVMAX - the maximum domestic sewer pipe velocity (m/s).
* SSVMAX - the maximum storm sewer pipe velocity (m/s).
* WVMIN - the minimum water velocity in a pipe (m/s).
* FRICTN - 'Manning's' pipe roughness coefficient.
* DSCOVR - minimum depth of coverage for domestic sewer pipe (m).
* SSCOVR - minimum depth of coverage for storm sewer pipe (m).
* SPAMAN - maximum spacing between manholes (m).
* SPALIT - maximum spacing between lights (m).

* USE - domestic watermain use (m^3/person/day).
* FLOAD - water consumption safety factor.
* ACCNUM - account number.
* DEFINE - full description of account.
* NACCNT - number of accounts.
* NUMELE - number of subdivision service elements.
* NACONT - number of account numbers associated with a subdivision element.
* ACCODE - account code.
* PRCENT - cost percentage of each subdivision service element by account.
* MBCOST - the unit cost of the subdivision element.
* MGRAD - maintenance cost gradient factor.
* MINST - the maintenance cost interest rate.
* MLIFE - the element's maintenance service life.
* OBCOST - the unit cost of the subdivision element.
* OGRATE - the operational cost gradient factor.
* OINST - interest rate for the operational costs.
* OLIFE - useful life of the element.

The INDAT1 routine obtains its input from the IOFILE routine, which when called by INDAT1 opens the economic data input file. INDAT1 then reads: the cost data into the COST_ variables listed above, and the other subdivision parameters into the rest of the variables listed above.

The cost data is divided into various sections that are differentiated by flags of -1 within the data files.

5.3.19 INDAT2- A Routine to Read the Subdivision Design Data

The variables used by INDAT2 are:

* RUNIT2 - unit device number as dictated by RCMAIN.
* NINT - total number of interest rate values.
* INTRST - interest rate value.
* LIFE - expected life of the project.
* NNODE - total number of nodes.
* NTYPE - type of node where: 1=termination
  2=directional
  3=3-way
  4=4-way
  5=other
  -3=access/egress
  -5=access/egress easements
* ELEV - ground elevation of the node.
* NLINK - total number of links.
* NODE1 - origin node of the link.
* NODE2 - destination node of the link.
* LLINK - link length.
* UNITS - number of single family units per link.
* MULTI - number of multi family units per link.
* AREA - total drainage area of link (Ha).

* LROAD - length of road on a given link (m).
* WROAD - width of road of a given link (m).
* TROAD - type of road/street.
* WWALK - sidewalk width (m).
* BARIER - barrier curb/gutter length per link (m).
* ROLLED - rolled curb/gutter length per link (m).
* BLVD - blvd. curb/gutter length per link (m).
* NWM - total number of destination links describing the watermain system.
* NDS - total number of destination links describing the domestic sewer system.
* NSS - total number of destination links describing the storm sewer system.
* RADIUS - radius of the cul-de-sac/tube (m).
* FRONT - saleable frontage per link (m).
* TBTRYW - locations (1-3) contain the origin links describing the watermain tributaries. Location 4 contains the destination link.
* TBTRYD - locations (1-3) contain the origin links describing the domestic sewer tributaries. Location 4 contains the destination link.
* TBTRYS - locations (1-3) contain the origin links describing the storm sewer tributaries. Location 4 contains the destination link.
* IQ - the total number of assigned an initial value, to follow:
* L - the number of the link.

* P - initial population of link for watermain calculation.
* PDS - initial population of link for domestic sewer calculation.
* ASS - initial population of link for storm sewer calculation.
* APARK - total area of the parks (m^2).
* ABUFF - total area of the buffers (m^2).
* ASCHOL - area of schools per subdivision (m^2).
* AMULT - area of multi family units per subdivision.
* ACOM - area of commercial section per subdivision (m^2).
* AOTHER - area of 'other' land per subdivision (m^2).
* ASUB - area of total subdivision (m^2).

The INDAT2 routine opens the input file by calling IOFILE. The link and node data is read into the arrays described by the variables above.

5.3.20 IOFILE- A Routine to Open the Files Used for Input and Output.

The variables used by IOFILE are:

* DEVICE - unit device name as dictated by RCMAIN.
* IST - flag indicating the naming of the input or output file.
* FNAME - the name of the input or output files.

The IOFILE's input is the device number under which to assign a unit number for the input and output files. The routine reads the user's names for these files and opens space for them on the VAX system.

5.3.21 **LIGHT - A Routine to Calculate the Number and Cost of Street Lights in the Subdivision.**

The variables used by LIGHT are:

* NNODE - number of nodes in subdivision.
* LTCOSN - light costs per node.
* LTNODE - number of lights per node.
* STRLIT - number of street lights per street length.
* NLINK - number of links in subdivision.
* LTCOSL - light costs per link.
* LTLINK - the number of lights per link.
* NTYPE - node type.
* CLINK - current link.
* TOTAL - street length between intersections and intersections,dead-ends.(temporary) (m).
* SLEN - street length of a given link (returned from CHECKLEN routine) (m).
* PATH - links between two intersections or intersection and dead end or intersection and cul-de-sac.
* STRLEN - street length between two intersections or intersection and dead end or intersection and cul-de-sac (m).
* ATTLIN - links associated with the specified node.
* NRGLIT - number of street lights.
* NSALIT - number cul-de-sac street lights.
* RADIUS - radius of the cul-de-sac/tube (m).
* WROAD - width of road of a given link (m).

* COST2 - street light cost.
* NBL - number of links between 2 intersections.
* BRANCH - number of links attached to the node.
* STLEN1 - the summation of the spacing between street lights (m).
* STLEN2 - the summation of the street lengths (m).
* EXRLEN - the extra street length attributed to the width of the road (m).
* SPALIT - the spacing distance between lights on a street (m).
* LIGHTS - the number of street lights on the current street.
* SLCOST - total cost of all street lights.
* NLIGHT - the total number of street lights.
* NLITS - the total number of street lights.
* NL - the number of links between two intersection,dead-end nodes.
* LINK1 - the link between type 2 node and the other nodes.

LIGHT calculates the number of street lights and their cost in the subdivision. First the routine identifies and counts the intersection and cul-de-sac nodes by calling COUNTER to count the number of 3 way, 4 way, cul-de-sac, and exit nodes in the subdivision. LIGHT then checks if the NTYPE is 3 or 4, if so each possible path leading from the node (there are 4) is processed by calling FINDPATH to identify the links between intersections, dead-ends, and

cul-de-sac nodes. If NL equals 0 processing continues as described in the third action paragraph. If NL is not 0 then CLINK is the assigned to the last link on the PATH.

Second, each link returned in the PATH is processed through CHECKLEN to calculate the length of cul-de-sacs, and TOTAL is incremented by SLEN. STRLEN of: the node being processed, one of the links attached to the node and the subsequent other node attached to this link, is assigned the value of TOTAL.

Third, LIGHT rearranges the STRLEN array and puts the subscripts in decreasing order. LIGHT checks STRLEN(N1,N2,J) in a triple do-loop incrementing each node, N1,N2 and J. If STRLEN(N1,N2,J) is not 0, FINDPATH is called, CLINK is assigned the value of the last link on the PATH, and each attached link is checked to see if it is equal to CLINK and it is identified as J2. If N1 is less than N2 and STRLEN(N2,N1,J2) equals 0 then STRLEN(N2,N1,J2) equals STRLEN(N1,N2,J).

Fourth, the street lights for the cul-de-sacs are calculated. Each node N1 is checked to see if its RADIUS is not 0 and NTYPE is 1. If so then NSALIT is incremented by 1, LTNODE(L1) equals 1, and LTCOSN equals COST2. Each ATTLIN (CLINK), is checked for a value of 0 and WROAD greater than 0. CLINK's origin and destination node types are checked to see if they are equal to 2. If so, CULCHK is called. If CUL equals 0, LTCOSL is incremented by COST2 and LTLINK is

incremented by 1. If CUL does not equals 0 LTCOSL(LINK1) is incremented by COST2 and LTLINK(LINK1) is incremented by 1. If neither of the origin or destination CLINK's Ntype's are 2 then LTCOSL(CLINK) is incremented by COST2 and LTLINK(CLINK) is incremented by 1.

If the NTYPE of the node is equal to 3 or 4, LTNODE and NRGLIT is incremented by the NTYPE of the node minus 2. LTCOSN equals LTNODE * COST2 and DMAKER is called. A loop is set to assign costs to the LINK that DMAKER finds. LTCOSL(LINK) is incremented by COST2 and LTLINK(LINK) by 1. Then each intersection is processed in a do-loop. Each node prior to the node being processed N2, assigns BRANCH to equal 4, the last node through the loop is the node being processed N1, and it assigns the value of 1 to BRANCH. As long as STRLEN of both of these nodes is not 0 and BRANCH is greater than 0 then BRANCH is decremented by 1. NBL is initialized to 0 and then incremented at a numbered command. CLINK becomes the NBLth link of the PATH between 2 intersections, an intersection and a dead-end, and an intersection and a cul-de-sac. NBL is incremented until CLINK of 0 is found at which point NBL is decremented and THREEWAY is called to identify the threeway intersection locations and to calculate their extra lengths.

The number of street lights per street length is calculated:

LIGHTS = INTEGER { [STRLEN + EXRLEN] / SPALIT - 0.5 }

NRGLIT is incremented by LIGHTS, and STRLIT(N1,N2) is equal to LIGHTS.

LIGHT determines the street that contains the lights and checks the length by adding the spacing between the lights. STLEN1-2 are initialized to 0 and EXRLEN respectively. CLINK is set to the beginning link of the PATH being processed starting at N1. If STLEN1 is Less than or equal to STLEN2 and LIGHTS is greater than 0, LTLINK(CLINK) is incremented by 1, LIGHTS is decremented by 1 and LTCOSL is incremented by COST2. STLEN1 is incremented by SPALIT until it is greater than STLEN2. The next link on the street is processed or the next CLINK is assigned. CHECKLEN is called to calculate the length of cul-de-sacs and STLEN2 is incremented by SLEN. The loop continues until all the intersections and PATHs are processed.

Finally the total number of street lights are calculated along with the costs of light for the subdivision. NLIGHT is incremented by NRGLIGHT,

SLCOST = NLIGHT * COST2

ASPLIT is called to divide and summarize subdivision costs amongst accounts. TTLCOST is incremented by SLCOST and the routine then outputs the results before passing control back to the RCMAIN.

5.3.22 MAINCE - A Routine to Calculate the Present and Future Values and Annuity Amounts of Subdivision Maintenance Costs

The variables used by MAINCE are:

* TYPE - subdivision element type number.
* TPCENT - check figure to ensure that 100% of an element's cost is allocated to accounts.
* NACONT - number of the account associated with a subdivision element.
* PRCENT - percentage of each element cost allocated to each account.
* METHOD - flag to indicate method to be used in allocating element costs to accounts. (If METHOD = 1; allocate by amount. If METHOD =2; allocate by percentage)
* MGRAD - maintenance cost gradient factor.
* GOGRAD - is a routine to calculate present,future values and annuity amounts.
* MGRATE - growth rate of the maintenance costs.
* MINST - the maintenance cost interest rate.
* MLIFE - the element's maintenance service life.
* P,A,F - present, future, annuity values.
* MNCOST - present, future, annuity values returned from GOGRAD, stored in an array.
* MNPACC - the amount of a subdivision element's cost allocated to an account.

* MCAPAF - present value and annual present value of the maintenance costs.
* NACCNT - number of accounts.
* ACCNUM - account number.
* ACCONT - the account name, associated with an element.
* ELEM - represents elements for each cost account.
* AMCOST - total maintenance costs by account.
* TIME - the number of times headings are to be overtyped on the output page.
* TMCOST - total maintenance costs for the subdivision.
* IMCOST - the annual maintenance costs of the subdivision elements.
* MNTFLG - flag indicating that MAINCE results should be written to the output file.
* MBCOST - the unit cost of the subdivision element.
* WMPIPE - the quantity of watermain pipe in the subdivision.
* DSPIPE - the quantity of domestic sewer pipe in the subdivision.
* SSPIPE - the quantity of storm sewer pipe in the subdivision (m).
* TPAREA - the quantity of pavement in the subdivision (m^2).
* TWK - the quantity of sidewalk area in the subdivision (m^2).

* TBR - the length of barrier curb and gutter in the subdivision (m).
* TRL - the length of rolled curb and gutter in the subdivision (m).
* TBL - the length of boulevard curb and gutter in the subdivision (m).
* APARK - the quantity of park area in the subdivision.(Ha.)
* ABUFF - the quantity of buffer area in the subdivision.(Ha.)
* NBASIN - the number of catch-basins in the subdivision.
* MNHOLE - the number of manholes in the subdivision.
* NLIGHT - the number of street lights in the subdivision.
* TSIGN - the number of street signs in the subdivision.
* NTRAFF - the number of traffic signs in the subdivision.
* TSING - the number of single-family dwellings in the subdivision.
* TMULT - the number of multi-family dwellings in the subdivision.

The MAINCE routine calculates maintenance costs and distributes them to the accounts. It also sums the costs by subdivision element.

First MAINCE determines the method by which costs are to be broken out. Each account is associated with, or charged a portion of, one or more of the construction element costs. MAINCE sums the percentage, PRCENT, of the element cost each account is associated with. If the summation, TPCENT, is less than or equal to 1, METHOD=1 and the percentage method is to be used. If TPCENT is greater than 1 the amount method is used.

Next the present value calculations are done by calling the GOGRAD routine and the ARITH(A,P,F) functions. (See page 291 for an explanation of GOGRAD and page 346 for an explanation of the functions and the financial formulas used in them.)

Following the present value calculations the correct method is used to accumulate the maintenance costs into the accounts. The various costs are multiplied by PRCENT and stored in MNPACC. The total per account is calculated and stored in AMCOST. The total maintenance cost per subdivision is calculated. The results are output by this routine.

5.3.23 **MANHOLE - A Routine to Calculate Engineering and Financial Data for Domestic and Storm Manhole Construction.**

The variables used by MANHOLE are:

* MANBED - height of the bedding beneath the manhole (m).
* NNODE - number of nodes in subdivision.
* DSCOSN - the cost of domestic manholes per node.
* MNHOLE - total number of manholes required in the subdivision.
* MHCOST - total cost of manholes in subdivision.
* SSCOSN - the cost of storm manholes per node.
* NLINK - number of links in subdivision.
* DSCOSL - the cost of domestic manholes per link.
* DSMAN - the number of domestic manholes per link.
* SSCOSL - the cost of storm manholes per link.
* SSMAN - the number of storm manholes per link.
* CUL - cul-de-sac node type 2 flag.
* NTYPE - node type.
* NODE1,N1 - origin node for the current link.
* NODE2,N2 - destination node for the current link.
* LINK1 - the link between type 2 node and other nodes.
* LENGTH - length of the link (m).
* CLINK - current link.
* LLINK - link length (m).

* HOLE - the number of manholes required per link per sewer.
* SPAMAN - spacing between catchbasins (m).
* ATTLIN - links associated with the specified node.
* DSELE1 - the domestic sewer elevation at the current link at the origin node (m).
* DSLINK - domestic sewer elevation of a given link (m).
* DSELE2 - the domestic sewer elevation at the current link at the destination node (m).
* ELEV1 - the elevation difference at the origin node (m).
* ELEV2 - the elevation difference at the destination node (m).
* ANG - the domestic sewer slope angle (rad).
* SSELE1 - the storm sewer elevation at the current link at the origin node (m).
* SSELE2 - the storm sewer elevation at the current link at the destination node (m).
* SSLINK - storm sewer elevation of a given link (m).
* COST15 - manhole shaft cost ($/m-depth).
* RADIUS - radius of the cul-de-sac/tube (m).
* WROAD - width of road of a given link (m).

The MANHOLE routine calculates the engineering and financial characteristics of the domestic and storm sewer manholes. The routine first initializes all cost accumulators (DSCOSL, SSCOSL, SSMAN, DSMAN) and manhole

counters (DSCOSN, SSCOSN). It then checks for the existence of cul-de-sacs depending on the node types of the CLINK. If NODE1 or NODE2 of CLINK is equal to 1 or 2, CULCHK is called to identify and calculate the length of the links leading to a cul-de-sac. And CLINK becomes LINK1 if CUL equals 1, otherwise CLINK equals the link being processed.

Secondly, HOLE, or the number of manholes required per link is calculated.

HOLE = INTEGER (LENGTH / SPAMAN - 0.5)

If HOLE is less than 1 the link is too short and is not assigned a domestic manhole.

Thirdly, the slope or ANG of the domestic sewer pipe is calculated by finding the elevation of the domestic sewer at the origin and destination nodes. It identifies which of the ATTLIN to the NODE1(CLINK) equal CLINK and sets DSELE1 equal to DSLINK(NODE1,ATTLIN) for the origin node. The same thing is then done for the destination node. Next MANHOLE checks to see if DSELE1,2 are both less than 9000.0. If so then MNHOLE is incremented by HOLE and DSMAN(CLINK) is assigned the value of HOLE. ELEV1,2 is equal to ELEV(NODE1,2) minus DSELE1,2. If ELEV1 is less than ELEV2, ANG equals:

ANG = ARCTAN [(ELEV2 - ELEV1) / LENGTH]

Fourth, the cost of each domestic sewer manhole is found by calculating the position of each manhole. The

position of each manhole is determined by using the spacing between manholes, the number of manholes and the slope of the domestic sewer pipe. Each manhole M is then used to calculate DSCOSL:

DPTH = TAN (ANG) * M * SPAMAN + ELEV1 + MANBED

DSCOSL(CLINK) = DSCOSL(CLINK) + COST15 * DPTH

If ELEV1 is less than ELEV2, ANG, DPTH, DSCOSL equal:

ANG = ARCTAN [(ELEV1 - ELEV2) / LENGTH]

DPTH = TAN (ANG) * (LENGTH - M * SPAMAN) + ELEV2 + MANBED

DSCOSL(CLINK) = DSCOSL(CLINK) + COST15 * DPTH

The total manhole link costs and subdivision totals are summed. DSCOSL is incremented by COST14 and MHCOST by DSCOSL.

Fifth, the storm sewer manhole data is calculated in a similar manner as the domestic sewer manholes as described above, starting from the paragraph describing the third action.

Sixth, each node is given a manhole and the cost is assigned to the node. If NTYPE of the node equals 2, then each ATTLIN NODE1 or NODE2 is checked to see if both the NTYPE equals 1 and RADIUS is not equal to 0. If so then CUL is set to 1 and processing continues. See the paragraph describing the eighth action.

If the check fails and CUL is 0, the lowest domestic pipe elevation is found. The DSLINK's of the node being processed ATTLIN's are checked to see if they fall between 9999 and 0. If they do DSELE1 equals DSLINK. If DSELE1 is less than 9000 MNHOLE is incremented by 1,

DPTH = ELEV - DSELE1 + MANBED,

DSCOSN = COST14 + COST15 * DPTH

and MHCOST is incremented by DSCOSN.

If the NTYPE of the node equals 7, the ATTLIN whose WROAD is not equal to 0 and is not itself equal to 0 is identified as SS. CULCHK is called to identify and calculate the length of the links leading to a cul-de-sac. If CUL equals 0 DMAKER is called to allocate subdivision elements to collector streets or links at major intersections. DSCOSL[LINK(1)] is incremented by DSCOSN and DSMAN[LINK(1)] is incremented by 1. If CUL is not equal to 0 DSCOSL(LINK1) is incremented by DSCOSN and DSMAN(LINK1) is incremented by 1. If NTYPE does not equal 7, DMAKER is called to allocate subdivision elements to collector streets or links at major intersections. DSCOSL[LINK(1)] is incremented by DSCOSN and DSMAN[LINK(1)] is incremented by 1.

Seventh, the storm sewer manholes are checked in exactly the same manner as the domestic sewer manholes were. See the paragraphs describing the sixth actions.

Eighth, the costs calculated are transferred to the appropriate accounts for cost allocation by ASPLIT. The manholes and their costs are output according to the links and nodes they are associated with. Control is then returned to the calling program.

5.3.24 OMINIT - A Routine to Calculate the Total Subdivision Element Costs

The variables used by OMINIT are:

* IMCOST - the annual maintenance costs of the subdivision elements.
* IOCOST - the annual operation costs of the subdivision elements.
* MBCOST - the unit cost of the subdivision element.
* WMPIPE - the length of watermain pipe in the subdivision (m).
* DSPIPE - the length of domestic sewer pipe in the subdivision (m).
* SSPIPE - the length of storm sewer pipe in the subdivision (m).
* TPAREA - the area of pavement in the subdivision (m^2).
* TWK - the length of sidewalk area in the subdivision (m).
* TBR - the length of barrier curb and gutter in the subdivision (m).
* TRL - the length of rolled curb and gutter in the subdivision (m).
* TBL - the length of boulevard curb and gutter in the subdivision (m).
* APARK - the park area in the subdivision (Ha.).

* ABUFF - the buffer area in the subdivision (Ha.).
* NBASIN - the number of catch-basins in the subdivision.
* MNHOLE - the number of manholes in the subdivision.
* NLIGHT - the number of street lights in the subdivision.
* TSIGN - the number of street signs in the subdivision.
* NTRAFF - the number of traffic signs in the subdivision.
* TSING - the number of single family dwellings in the subdivision.
* TMULT - the number of multi family dwellings in th subdivision.

The IMCOST and IOCOST arrays are filled the results of multiplying MBCOST and OBCOST by the other variables listed above (eg. WMPIPE * MBCOST(1)=IMCOST).

5.3.25 OPERAT - A Routine to Calculate the Present and Future Values and Annuity Amounts of Subdivision Operational Costs

The variables used by OPERAT are:

* TYPE - subdivision element type number.
* TPCENT - check figure to ensure that 100% of an element's cost is allocated to accounts.
* NACONT - number of account number associated with a subdivision element.
* PRCENT - percentage of each element cost allocated to each account.
* METHOD - flag to indicate method to be used in allocating element costs to accounts.
* OGRAD - the operational costs that will escalate the operational costs every year.
* GOGRAD - is a routine to calculate present,future values and annuity amounts.
* OGRATE - growth rate of the operational costs.
* OINST - the operational cost interest rate.
* OLIFE - the element's operational service life.
* P,A,F - present, future, annuity values.
* OPCOST - present, future, annuity values returned from GOGRAD, stored in an array.
* OPPACC - the amount of a subdivision element's cost allocated to an account.
* OCAPAF - present value and annual present value of the operational costs.

* NACCNT - the number of accounts.
* ACCNUM - account number.
* ACCONT - the account name, associated with an element.
* ELEM - represents elements for each cost account.
* AOCOST - total operational costs by account.
* TIME - the number of times headings are to be overtyped on the output page.
* TOCOST - total operational costs for the subdivision.
* IOCOST - the annual operational costs of the subdivision elements.
* OPRFLG - flag indicating that OPERAT results should be written to the output file.
* OBCOST - the unit cost of the subdivision element.
* WMPIPE - the quantity of watermain pipe in the subdivision (m).
* DSPIPE - the quantity of domestic sewer pipe in the subdivision (m).
* SSPIPE - the quantity of storm sewer pipe in the subdivision (m).
* TPAREA - the quantity of pavement in the subdivision (m^2).
* TWK - the quantity of sidewalk area in the subdivision (m^2).
* TBR - the length of barrier curb and gutter in the subdivision (m).

* TRL - the length of rolled curb and gutter in the subdivision (m).
* TBL - the length of boulevard curb and gutter in the subdivision (m).
* APARK - the quantity of park area in the subdivision (Ha.)
* ABUFF - the quantity of buffer area in the subdivision.(Ha.)
* NBASIN - the number of catch-basins in the subdivision.
* MNHOLE - the number of manholes in the subdivision.
* NLIGHT - the number of street lights in the subdivision.
* TSIGN - the number of street signs in the subdivision.
* NTRAFF - the number of traffic signs in the subdivision.
* TSING - the number of single family dwellings in the subdivision.
* TMULT - the number of multi family dwellings in the subdivision.

The OPERAT routine calculates operational costs and distributes them to the accounts. The routine also sums the costs by subdivision element.

First OPERAT determines the method by which costs are to be broken out. Each account is associated with, or

charged a portion of, one or more of the construction element costs. OPERAT sums the percentage, PRCENT, of the element cost each account is associated with. If the summation, TPCENT, is less than or equal to 1, METHOD=1 and the percentage method is to be used. If TPCENT is greater than 1, the amount method is used.

Next present value calculations are done by calling GOGRAD and the ARITH(A,P,F) functions. (See page 291 for an explanation of GOGRAD and page 346 for an explanation of the functions and the financial formulas used in them.).

Following the present value calculations the correct method is used to accumulate the operational costs into the accounts. The various costs are multiplied by PRCENT and stored in OPPACC. The total per account is calculated and stored in AOCOST. The total operational cost per subdivision is calculated. The results are output by this routine.

5.3.26 OUTDAT1 - A Routine to Output the Unit Cost Data for the Service Elements.

The variables used by OUTDAT1 are:

* COST1 - street sign unit cost ($/sign).
* COST2 - street light cost ($/light).
* COST3 - park development ($/hectare).
* COST4 - buffer development ($/hectare).
* COST5 - collector street costs ($/$m^2$).
* COST6 - local street costs ($/$m^2$).
* COST7 - cost of sidewalk ($/$m^2$).
* COST8-10 - unit cost of barrier, rolled, boulevard curb and gutter ($/m).
* COST11-13 - watermain, domestic and storm sewer costs ($/m).
* COST14 - manhole - base, cover, riser, frame ($/manhole).
* COST15 - manhole shaft cost ($/m-depth).
* COST16 - catchbasin - base, cover, riser, frame ($/catchbasin).
* COST17 - catchbasin shaft ($/m-depth).
* COST18 - catchbasin lead costs ($/m).
* COST19 - single-family unit power costs ($/unit).
* COST20 - multi-family unit power costs ($/unit).
* COST21 - single-family unit gas costs ($/unit).
* COST22 - multi-family unit gas costs ($/unit).

* COST23 - single-family unit phone costs ($/unit).
* COST24 - multi-family unit phone costs ($/unit).
* COST25 - street sweeping costs ($/km/yr).
* COST26 - winter road costs ($/km/yr).
* COST27 - asphalt maintenance ($/km/yr).
* COST28 - concrete maintenance ($/km/yr).
* COST29 - sewer maintenance ($/km/yr).
* COST30 - water distribution costs ($/km/yr).
* COST31 - solid waste collection ($/hectare/yr).
* COST32 - street light maintenance ($/light/yr).
* COST33 - park and buffer maintenance ($/hectare/yr).

The OUTDAT1 routine outputs all of the service element unit cost data with labels.

5.3.27 OUTDAT2- A Routine to Output the Node and Link Design Data

The variables used by OUTDAT2 are:

* NNODE - total number of nodes.
* NTYPE - type of node.
* ELEV - ground elevation of the node (m).
* NLINK - total number of links.
* NODE1 - origin node of the link.
* NODE2 - destination node of the link.
* LLINK - link length (m).
* UNITS - number of single family units per link.
* MULTI - number of multi family units per link.
* LROAD - length of road on a given link (m).
* WROAD - width of road of a given link (m).
* TROAD - type of road/street.
* WWALK - sidewalk width (m).
* BARIER - barrier curb/gutter length per link (m).
* ROLLED - rolled curb/gutter length per link (m).
* BLVD - blvd. curb/gutter length per link (m).
* WUNIT1 - the output device number code.

The OUTDAT2 routine outputs the subdivision design data referred to above. The data is written to the device specified by WUNIT1.

5.3.28 **PARKBUFF - A Routine to Calculate Park and Buffer Costs**

The variables used by PARKBUFF are:

* PCOST - total cost of park development.
* BCOST - total cost of buffer development.
* TLCOST - total subdivision cost.

PARKBUFF determines the development costs of the park and buffer areas. PCOST and BCOST are calculated by multiplying the corresponding areas by their respective unit costs. ASPLIT is called to categorize these costs according to their account numbers. TLCOST is updated by adding PCOST and BCOST. The results are output to the device specified by WUNIT.

5.3.29 PAVEMENT - A Routine to Calculate the Area of Pavement and Allocate the Area Costs to the Appropriate Links

The variables used by PAVEMENT are:

* TAREA - total area of the pavement (m^2)
* PI - arc cosine of -1.0 (rad).
* TLEN - total length of the pavement (m).
* LOCLEN - total local street pavement length (m).
* COLLEN - total collector street pavement length (m).
* ACOL - total collector street pavement area (m^2).
* ALOC - total local street pavement area (m^2).
* NLINK - number of links in subdivision.
* CNODE - current node.
* NODE1 - origin node for the current link.
* NTYPE - node type.
* WD - road width (from ADJ routine) (m).
* I - current link (for ADJ routine).
* CORWD - road width of a given link (m).
* NODE2 - destination node for the current link.
* WROAD - width of road of a given link (m).
* RADIUS - radius of the cul-de-sac/tube (m).
* PNODE - previous node in memory.
* ASAC - area of the road (from COR/SAC/TUBE routines) (m^2).

* LEN - road length (from COR/SAC/TUBE routines) (m).
* ALINK - area of the road on a link (m^2).
* LROAD - calculated length of road on link (m).
* RDPINT - road length per link (m).
* LLINK - link length (m).
* TV - length of a link less the width of half the intersections at the ends of the road (m).
* BULB - the number of bulb curves.
* EXTRA - the length of the link within the bulb curve (m).
* AINTS - area of the intersection (m^2).
* CURVE - the area in front of the sidewalk curb (m^2).
* LINK - the link to which elements are charged to (returned from DMAKER).
* RCURB - radius of the corner curb (m).
* DIRECT - indicates the origin/destination node for a given link.
* ACOL - total collector street pavement area (m^2).
* COLLEN - total collector street pavement length (m).
* PVCOST - cost of the pavement area per link ($\$/m^2$).
* LOCLEN - total local street pavement length (m).
* COST6 - local street costs ($\$/m^2$).
* COST5 - collector street costs ($\$/m^2$).
* CSCOST - total collector street pavement cost.
* LSCOST - total local street pavement cost.

* TLCOST - total subdivision pavement cost.

The PAVEMENT routine calculates the pavement areas and costs for the local, collector, bulb curve and intersections, and allocates the area to the local and collector streets. It also calculates the cost and area per link.

First PAVEMENT determines the width of the various links by calling ADJ to determine the links attached to NODE1 and NODE2, and the width of the corresponding road. The CORWD of the link being processed is assigned the value of WD returned from ADJ only if the NTYPE of the road is a 3 or 4. If the NTYPE check fails then CORWD equals zero signifying an easement.

Second, PAVEMENT checks each link's WROAD to determine if they are 0, if so PAVEMENT skips to the paragraph describing PAVEMENT's sixth action.

If the NTYPE of the NODE1 of the link is 1, CNODE becomes the destination node and PNODE the origin NODE. If NTYPE of NODE2 is 1, PNODE becomes the destination node and CNODE the origin NODE1. Next PAVEMENT checks if CNODE has an NTYPE of 2. If not this indicates a tube sac and PAVEMENT checks for a one by checking RADIUS(PNODE), if RADIUS(PNODE) is not equal to 0 TUBE is called and ALINK is given the value of ASAC, LROAD the value of LEN and RDPINT the value of LROAD. If RADIUS(PNODE) is equal to 0:

TV = LLINK(I) - [CORWD(I,1) + CORWD(I,2)] / 2

LROAD is assigned the value of TV, ALINK equals TV * WROAD and RDPINT equals LROAD. If the check for the tube sac initially fails and the NTYPE(CNODE) is 2, PAVEMENT checks the RADIUS(PNODE). If it is not equal to 0 a cul-de-sac is present and SAC is called. ALINK is assigned the value of ASAC, LROAD equals LEN, and RDPINT equals LROAD. If RADIUS(PNODE) is equal to 0, indicating no cul-de-sac, ALINK equals LLINK * WROAD, LROAD equals LLINK, and RDPINT equals LROAD.

Third, the straight road and bulb curve areas are calculated. The bulb curves are first identified by checking if the NTYPE's of NODE1,NODE2 both equal 2 and RADIUS(NODE1) does not equal 0. CNODE equals NODE1 and BULB equals 1.

If the NTYPE's of NODE1,NODE2 both equal 2 and RADIUS(NODE2) does not equal 0 then CNODE equals NODE2 and BULB is incremented by 1.COR is called for the area calculation.

If the NTYPE's of NODE1,NODE2 both equal 2 and BULB does not equal 0 then COR is called to calculate the pavement area for the bulb curves. ALINK is assigned the value of ASAC, LROAD equals LEN, RDPINT equals LROAD. If BULB equals 0, indicating no bulb curves, ALINK equals LLINK * WROAD, LROAD equals LLINK, and RDPINT equals LROAD.

Fourth intersection areas are calculated. If the NTYPE of NODE1 is 2, PNODE becomes the NODE1 the origin node and CNODE the destination NODE2. If the NTYPE(CNODE) equals 3 or 4, and RADIUS(PNODE) is not equal to 0, BULB equals 1 and COR is called to calculate the pavement area for straight roads streets and bulb curves. EXTRA, ALINK, LROAD are calculated:

EXTRA = [CORWD(I,1) + CORWD(I,2)] / 2.0

ALINK = ASAC - WROAD(I) * EXTRA

LROAD = LEN - EXTRA

and RDPINT equals LROAD.

If RADIUS(PNODE) is equal to 0, LROAD is assigned the value of TV, ALINK equals TV * WROAD and RDPINT equals LROAD.

TV = LLINK(I) - [CORWD(I,1) + CORWD(I,2)] / 2

Fifth, PAVEMENT checks if the NTYPE of NODE1 and NODE2 equal 3 or 4, indicating an intersection crossing an intersection. If so:

LROAD = LLINK - [CORWD(I,1) + CORWD(I,2)] / 2.0

ALINK = LROAD * WROAD and RDPINT equals LROAD.

Sixth, PAVEMENT calculates CURVE:

CURVE = $RCURB^2$ * (1.0 - 0.25 * PI).

The intersection area is calculated and is split amongst the collector streets by calling DMAKER. If the NTYPE of each node is 3 the variable A equals 2 otherwise A is equal to 4. DMAKER is called to allocate the pavement area to collector streets at major intersections. If the node being processed is an exit node, then:

CLINK = LINK(1) returned from DMAKER

N = the node being processed.

AINTS = [WROAD(CLINK)^2]/2.0 + 2.0*CURVE

ALINK(CLINK) is incremented by AINTS(N), and RDPINT is incremented by 0.5*CORWD(CLINK,DIRECT).

If there is no exit node, PAVEMENT checks if NODE1(CLINK) equals N and DIRECT equals 1; if it does not equal N, DIRECT equals 2. AINTS equals:

AINTS(N) = WROAD(CLINK)*CORWD(CLINK,DIRECT) + A*CURVE

ALINK(LINK1,2) is incremented by AINTS(N)/2.0 and RDPINT is incremented by 0.5*CORWD(CLINK,DIRECT).

The areas and costs are summed by link and then by collector and local streets and by subdivision. If TROAD equals 1 ACOL is incremented by ALINK, COLLEN by LROAD, and:

PVCOST = ALINK*COST5

If TROAD does not equals 1 ALOC is incremented by ALINK, LOCLEN by LROAD, and:

PVCOST = ALINK*COST6

The routine outputs the pavement lengths areas and costs.

TAREA = ALOC + ACOL

TLEN = COLLEN = LOCLEN

CSCOST = ACOL * COST5

LSCOST = ALOC * COST6

TLCOST = TLCOST + CSCOST + LSCOST

Control is then passed back to the calling routine.

5.3.30 SIDEWALK - A Routine to Calculate Sidewalk Area and Cost for the Subdivision

The variables used by SIDEWALK are:

* WIDTH - minimum sidewalk width (m).
* TWK - total sidewalk area(m^2).
* TBR - total barrier curb and gutter length (m).
* TRL - total rolled curb and gutter length (m).
* TBL - total boulevard curb and gutter length (m).
* SRCOST - total sidewalk cost per link.(barrier, rolled, boulevard)
* WWALK - sidewalk width (m).
* ASIDE - sidewalk area (m^2).
* LWALK - sidewalk length (m).
* TROAD - type of road or street.
* BARIER - barrier curb/gutter length per link (m).
* ROLLED - rolled curb/gutter length per link (m).
* BLVD - blvd. curb/gutter length per link (m).
* COST7-10 - unit cost of sidewalk, barrier, rolled, boulevard curb and gutter.
* WKCOSL - link cost of sidewalk.
* BRCOSL - link cost of barrier curb and gutter.
* RLCOSL - link cost of rolled curb and gutter.
* BLCOSL - link cost of boulevard curb and gutter.
* TLCOST - total cost of sidewalk (including curb and gutter) for subdivision.

* WKCOST - sidewalk element cost per subdivision.
* BRCOST - barrier curb and gutter element cost per subdivision.
* RLCOST - rolled curb and gutter element cost per subdivision.
* BLCOST - boulevard curb and gutter element cost per subdivision.

The SIDEWALK routine calculates the link and subdivision cost of the construction of the sidewalk and curbs and gutters.

The routine first checks whether the link has a sidewalk, by checking if WWALK equals 0, if affirmative calculates the sidewalk area by multiplying LWALK of the link by WIDTH(TROAD) of the link. If WWALK does not equal 0 then ASIDE equals WWALK * LWALK.

Second, the areas and widths of the curbs, gutters and sidewalk are summed for the subdivision (TWK,TBR,TRL,TBL).

The link costs are calculated for the elements and in total for the link. Then the elemental costs are summed. The elemental costs are allocated to the various accounts by calling ASPLIT. The total subdivision cost is calculated and the elemental costs and areas are output.

5.3.31 SIGN - A Routine to Determine the Street and Traffic Sign Requirements for the Links and Nodes

The variables used by SIGN are:

* NSTR - number of street signs in subdivision.
* TYELD - number of yield signs in subdivision.
* TSTOP - number of stop signs in subdivision.
* YCOST - cost of yield signs in subdivision.
* SCOST - cost of stop signs in subdivision.
* NYELD - number of yield signs per node.
* NNODE - number of nodes in subdivision.
* NSIGN - number of street signs per node.
* SGCOST - cost of street signs per link.
* LINK - stores the collector/major local link (returned from DMAKER).
* NCOL - number of collector streets.
* NLOC - number of local street.
* CLINK - current link.
* ATTLIN - links associated with the specified node.
* WROAD - width of road of a given link (m).
* TROAD - type of road/street.
* LYELD - number of yield signs per link.
* YDCOST - yield sign cost per link.
* COST1 - street sign unit cost.
* LSTOP - number of stop signs per link.
* STCOST - stop sign cost per link.

* TFCOST - total traffic sign cost for subdivision.
* NTRAFF - number of traffic signs in subdivision.
* SNCOST - street sign cost for subdivision.

The SIGN routine calculates the number and cost of street and traffic signs in the subdivision. The routine allocates one street sign per intersection. At threeway intersections local streets crossing collectors require a yield sign, collectors crossing collectors require a stop sign. At fourway intersections, local streets crossing collectors require 2 yield signs, collectors crossing collectors require 2 stop signs.

If the NTYPE of the node is 3 or 4, NSTR is incremented by 1. DMAKER is called to identify the major LINK. NSIGN(LINK) is incremented by 1, SGCOST(LINK) is incremented by COST1.

Secondly the ATTLIN of the node being processed are checked to see if they are equal to 0 and WROAD is 0. If it is, processing continues, if not TROAD of the ATTLIN being processed is checked to see if it is equal to 1. If so NCOL is incremented by 1, otherwise NLOC is incremented by 1.

Third, if NCOL is greater than or equal to 2 local streets crossing collectors are processed. If NCOL is less than 2 SIGN commences its sixth process. If NCOL equals 2 and NLOC is less than 2, then NYELD equals NTYPE minus 2,

and DMAKER2 is called to assign subdivision elements to the local street LINK. LYELD(LINK) is incremented by 1, YDCOST and YCOST are incremented by COST1.

Fourth, if NCOL is greater than or equal to 3 and NLOC equals 0, collector streets crossing collector streets are being processed. NSTOP equals NTYPE minus 2 of the node being processed and DMAKER2 is called. LSTOP(LINK) is incremented by 1, STCOST and SCOST are incremented by COST1.

Fifth, if NCOL equals 3 and NLOC is not 0, then NSTOP equals 2. DMAKER2 is called and each stop sign is allocated to LINK(1) and LINK(2) as identified by DMAKER2. LSTOP(LINK)is incremented by 1, STCOST and SCOST are incremented by COST1.

Sixth, if NCOL + NLOC is greater than or equal to 3 and NCOL does not equal 0, collector streets at uneven crossings are being processed. NYELD equals NTYPE minus 2 of the node being processed and DMAKER2 is called. LYELD(LINK) is incremented by 1, YDCOST and YCOST are incremented by COST1.

Seventh, the total number of traffic signs required are calculated by summing the NYELD, NSTOP by node.

TFCOST = YCOST + SCOST

NTRAFF = TYELD + TSTOP

SNCOST = NSTR * COST1

Eighth, ASPLIT is called to allocate the street and traffic signs costs to the appropriate accounts. The sign costs and amounts are output and control is returned to the main routine.

5.3.32 STORM - A Routine to Design the Storm Sewer System and Find the Associated Costs

The variables used by STORM are:

* SSDPTH - minimum depth of storm sewer (m).
* SSCOST - total cost of storm sewer.
* TSAREA - total drainage area of each destination link (including upstream area) (Ha.)
* SSLOPE - slope of storm sewer from origin node to destination node.
* SSELEV - elevation of storm sewer pipe (m).
* AS - total drainage area of each link (Ha).
* QSS - total generated storm sewer flow (m^3/s).
* D - calculated pipe diameter (m).
* D3 - required diameter of storm sewer (m).
* DIASS - diameter of storm sewer (m).
* INDX3 - diameter index for storm sewer (1-14).
* DINDX3 - depth index for storm sewer (1-9).

STORM designs the storm sewer system and estimates the associated costs. For each link, the routine estimates the generated storm sewer flow, calculates the slope and the pipe size required to carry the estimated sewer flow, selects a standard sized storm sewer pipe based on the calculated diameter, verifies the maximum and minimum allowable water velocities, adjusts the pipe slope and/or pipe size to meet the tolerances set by the maximum and

minimum allowable water velocities, and calculates the average depth of ground cover over the storm sewer.

First, STORM outputs the minimum and maximum acceptable water velocities, the minimum depth of ground over the sewer pipe and a header describing the output data. Then a loop indexes each group of storm sewer tributary data. A second loop indexes each feed link in the tributary data group. A third loop is used to compare the receptor links of other tributary data groups to the indexed link. If a match is found, TOLINK is set to one. Third, the expected storm sewer flow is calculated using a standard engineering formula obtained from the City of Regina Public Works and Engineering Department.

Third the area contributing to the storm sewer flow for the indexed link is calculated by summing the area for the indexed link and the total upstream area. (The upstream area was combined with the original initial area values for the link ASS: Remember, the initial area values represent the potential contributing storm runoff area that the link must convey support.) The total area for the link is added to TSAREA, which will eventually contain the total area of all the feed links contributing flow to the 'from' links of the indexed tributary data group. The expected storm sewer flow is calculated next using formulas derived from data obtained from the City of Regina. The existing slope of the pipe is calculated and used in Manning's formula to calculate the

pipe diameter. This diameter is rounded up to the nearest standard pipe size using a series of IF-THEN statements.

Next, DMSTC calls DIAMTR to determine whether the calculated slope and the chosen pipe size meet the maximum and minimum water velocity limits. If DIAMTR is unable to adjust the slope with the chosen pipe size so that the water velocity falls within the defined tolerances the next largest pipe diameter is selected. The minimum trenching depth is determined by summing the minimum depth of ground cover, and the pipe diameter and adding 0.1 meters. The trenching elevation of each end of the pipe is found by subtracting the minimum trenching depth from the 'from' node and using the link length and the slope to calculate the elevation at the 'to' node. The routine ensures that the 'from' node of the next 'to' link has the lowest elevation possible if more than one 'from' link contributes flow to the 'to' link.

The cost of installing the storm sewer pipe is calculated by multiplying the link length by the unit cost. The unit cost is a function of the diameter index-IND*3 and the depth Index -DIND*3. The cost of the entire storm sewer system is a summation of the storm sewer costs for each link. The diameter is converted from meters to millimeters and the link I.D., feed area, flow, ground slope, diameter, fluid velocity, pipe slope, depths of pipe ends, average depth and depth index, are output to the specified device.

The excavation depth is saved, with a call to SSDPTH. Upon completion of the second loop all upstream areas for the most recent tributary group are added to the initial conditions of the receptor link. The first loop continues until all the groups have been analyzed.

Finally, ASPLIT is called to assign the storm sewer system costs to the appropriate account. Control then return to the calling program.

5.3.33 UTILITY - A Routine to Calculate Utility Costs

The variables used by UTILITY are:

* TMULTI - total number of multi family units.
* TSNGLE - total number of single family units.
* TMCOST - total cost of utilities for multi-family units.
* TSCOST - total cost of utilities for single family units.
* PWSING - single family power costs per link.
* GSSING - single family gas costs per link.
* PHSING - single family telephone costs per link.
* PWMULT - multi-family power costs per link.
* GSMULT - mutli-family gas costs per link.
* PHMULT - multi-family phone costs per link.
* UTCOST - total utility cost per link.
* TLCOST - total subdivision cost.

UTILITY calculates the telephone, gas and power costs for all multi- and single-family units in the subdivision. A loop indexes each link. Then, TMULTI and TSNGLE are calculated by summing the number of multi- and single-family units, respectively, on each link. PSWING, GSSING, PHSING, PWMULT, GSMULT, and PHMULT are calculated for each link by multiplying the corresponding number of single- or multi-family units by the respective unit cost (COST19 to COST24). UTCOST for each link is determined by summing PSWING,

GSSING, PHSING, PWMULT, GSMULT and PHMULT. Upon completion of the loop, TMCOST and TSCOST are calculated by multiplying the total number of single and multi-family units by their respective units costs. TLCOST is updated by adding TMCOST and TSCOST. The results are then output to the device specified by WUNITL.

5.3.34 WATER - A Routine to Design the Watermain System and Estimate the Associated Costs

The variables used by WATER are:

* USE - expected domestic water use per person (m^3/PERSON/DAY).
* FLOAD - safety factor for water requirements to account for excessive system demands during fire fighting and peak draw periods.
* CMMCMS - conversion factor from cubic meters per minute (CMM) to cubic meters per second (CMS).
* WMCOST - total watermain cost.
* TPOP - total population feeding each destination link.
* P1 - total population feeding each link.
* QDMSTC - domestic water flow requirement (m^3/min.)
* QFIRE - flow requirement for fire fighting purposes (m^3/min.)
* QWM - total water flow requirement for the watermain.
* D1 - diameter of current pipe (m).
* DIAWM - diameter of current pipe (mm).
* CWM - per unit cost of watermain in each link.
* INDX1 - diameter index for watermain (1-6).

WATER designs the watermain system and estimates for associates costs.

First, WATER send a header describing the output data to the device specified by WUNIT. Second, a loop indexes each group of watermain tributary data. A second loop indexes each feed link in the tributary data group.

The number of people consuming water from the indexed link is determined by summing the number of single- and multi- family house dwellers on the link and the total number of people upstream (the number of upstream consumers is combined with the original initial population values for the link in P: Remember, the initial population values represent the potential number of people on a link who do not live on the link but will consume water while shopping or attending school, etc.) The total number of water consumers for the link is added to TPOP, which will eventually contain the total number of people consuming water for a particular group of tributary data. TPOP is purged prior to analyzing a new tributary data group. The domestic and fire fighting water requirements are calculated using formulas obtained from Hicks[1].

The summation of QDMSTC and QFIRE is used to calculate the required pipe diameter. The link population, flow and diameter are then output to the specified device. The calculated pipe diameter is rounded up to the nearest standard pipe size using a series of IF-THEN statements. The

cost of installing the watermain for the indexed link is calculated by multiplying the link length by the unit cost. The unit cost is a function of the diameter index (INDX1). The cost of the entire watermain system is a summation of the watermain costs per each link. The loop returns to the next element in the group. When all group elements have been analyzed, TPOP is added to the initial population of the destination link so that the upstream population will be incorporated in the downstream links. When all tributary groups have been indexed, ASPLIT is called to assign the watermain costs to the appropriate account. Control then returns to the calling routine.

$$Q_D = [(Use)(POP)F]/(24 * 60)$$

$$Q_F = 1020*0.003785[(POP/1000)(1-0.01(POP/1000)^{1/2})]^{1/2}$$

Where:

Q_D = domestic flow requirement.

Q_F = fire flow requirement.

Use = expected domestic water use

F = safety factor

Pop = population

The above equations are taken from:

1. Standard Handbook of Engineering Calculations, New York, 1972, by Tyler Hicks.

5.4 SUBROUTINES CALLED BY OTHER SUBROUTINES

5.4.1 ACCOUT - A Routine to Calculate Construction Costs.

The variables used in ACCOUT are:

* TIME - number of times to overprint headings.
* NACONT - number of account number associated with a subdivision element.
* TYPE - subdivision element type number.
* INTRST - interest rate in percentage.
* LIFE - the life of the construction element.
* SDCOST - the capital amount to be adjusted by the present value factor and the rate.
* ACCONT - the account name, associated with an element.
* AWORTH - the annual expenditure to be discounted back to present values.
* TYCOST - present cost sorted by element and account.
* TANNUL - present annual cost sorted by element and account.

ACCOUT calculates several construction cost values and outputs them with the construction cost breakdown. ACCOUNT is the calling program for this routine.

First, ACCOUT calculates the annual worth, AWORTH, the future worth, FWORTH, and the total annual cost, TANNUL, of each subdivision element by account number in a loop. AWORTH and FWORTH are calculated by passing the arguments INTRST,

LIFE and SDCOST to the functions UNFORM and UNFORM2 respectively. These functions perform the desired calculations and return the annual worth of construction costs in the case of UNFORM and the future worth of construction costs in the case of UNFORM2. (See page 348, on UNFORM and UNFORM2 for details of calculations.) TANNUL is calculated by summing the AWORTH values for all the account numbers for each element.

Secondly, ACCOUT outputs the account number, ACCONT, present construction cost, TYCOST, and the total annual cost, AWORTH, for each subdivision element by account number. This output is done using the same loop in which the above output calculations are done.

Thirdly, ACCOUT outputs a total for present construction costs, TYCOST, and a total for annual construction costs, TANNUL. This output is overtyped to highlight it, the number of overstrikes being determined by TIME. Control is then returned to the calling program.

5.4.2 ACCOUT1 - A Routine to Output a Section of the Maintenance Cost Breakdown

Variables used by ACCOUT1 are:

* NACONT - number of account number associated with a subdivision element.
* TYPE - the subdivision element type number.
* ACCONT - name of the account associated with a subdivision element.
* MCAPAF - present and annual maintenance costs.
* J,K - loop index variables

ACCOUT1 outputs maintenance cost values with the maintenance cost breakdown. ACCOUNT1 is the calling program for ACCOUT1.

ACCOUT1 uses a loop to output the account number, ACCONT, the present maintenance cost, MCAPAF(TYPE,J,1), and the annual maintenance cost, MCAPAF(TYPE,J,2). The two cost outputs are within an implied do-loop.

5.4.3 ACCOUT2 - A Routine to Output a Section of the Operation Costs Breakdown

Variables used by ACCOUT2 are:

* NACONT - number of account numbers associated with a subdivision element.
* TYPE - subdivision element type number.
* ACCONT - account number associated with a subdivision element.
* OCAPAF - present and annual operation costs.

ACCOUT2 outputs operation cost values with the operation cost breakdown. ACCOUNT2 is the calling program for ACCOUT2.

ACCOUT2 uses a loop to output the account number, ACCONT, the present operation cost, OCAPAF(TYPE,J,1), and the annual maintenance cost, OCAPAF(TYPE J,2). The two cost outputs are within an implied do-loop.

5.4.4 ADJ - A Routine to Determine the Links Attached to Each Node and the Width of the Corresponding Road

The variables used by ADJ are:

* ATTLIN - links associated with the current node.
* NLINK - number of links.
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.
* CNODE - current node in memory.
* NL - number of links attached to current node.
* CLINK - current link in memory.
* EXIT - a flag indicating the exit node.
* CWD - width of corresponding road.
* WROAD - width of road of a given link (m).

The ADJ routine calculates the number of links attached to each node. Each link is checked to determine the nodes attached to it. If the node attached to a link corresponds to the node currently being checked, then ATTLIN stores the node and link numbers.

When ADJ is called by PAVEMENT, then it determines the width of the link that is being crossed, by the link being passed. The width of the road, WROAD, of all the attached links, ATTLIN, at the current node, CNODE, is compared to the WROAD of the CLINK. If the WROAD of the ATTLIN is the same as that of the CLINK, CWD = WROAD. If WROAD is

different, then CWD = WROAD of the link with a different WROAD. Control is then returned to PAVEMENT.

5.4.5 ASPLIT - A Routine to Divide and Summarize Subdivision Costs Amongst Accounts

The variables used by ASPLIT are:

* TPCENT - check figure to ensure that 100% of an element's cost is allocated to accounts.
* PRCENT - percentage of each element cost allocated to each account.
* NACONT - number of accounts associated with each element.
* TYPE - subdivision element type.
* METHOD - flag to indicate method to be used in allocating element costs to accounts.
* SDCOST - cost allocated to account from total elemental construction cost.
* TYCOST - total subdivision element construction cost.
* NACCNT - number of accounts.
* ACCNUM - account number.
* ACCONT - the account name, associated with an element.
* ELEM - represents elements for each cost account.
* ACCOST - account total cost.

The ASPLIT routine distributes construction costs to the accounts. It also sums the costs by subdivision element.

First ASPLIT determines the method by which costs are to be broken out. Each account is associated with, or charged a portion of, one or more of the construction element costs. ASPLIT sums the percentage, PRCENT, of the element cost each account is associated with. If the summation, TPCENT, is less than or equal to 1, METHOD=1 and the percentage method is to be used. If TPCENT is greater than 1 ,METHOD=2,the amount method is used.

The correct method is used to accumulate the maintenance costs into the accounts. The various costs are multiplied by PRCENT and stored in SDCOST. The total per account is calculated and stored in ACCOST. The total construction cost per subdivision is calculated. The results are output by this routine.

5.4.6 CATCALC - A Routine to Calculate Catchbasin Construction Costs for a Node

The variables used by CATCALC are:

* CLINK - current link in memory.
* ATTLIN - links associated with the given node.
* WROAD - width of road of a given link (m).
* AVEDEP - average storm sewer elevation for a given link (m).
* IND - temporary depth index value.
* INDEX - depth index for catch-basin.
* BASIN - number of catch-basins required for the current node. BASIN is passed from CATNODE for node types 1 and 2, and from CATBAS for node types 3 and 4.
* CNODE - current node in memory.
* FLINK - flag indicating the drainage link.
* CBCOS - cost of catch-basins for the node.
* COST16-18 - elemental unit costs.
* CBLEAD - length of the catch-basin lead (m).
* LBASIN - number of catch-basins required for the current link.
* LCBCOS - cost of the catch-basins for the current link.

* CBC - cost for a complete catch-basin at a given node, including the base, cover, riser, shaft and lead costs.
* SS - subscript for CBLEAD used in calculating CBC.
* EXL - the link with one extra catch basin.
* CLEAD - minimum length of the catch-basin lead (m).

CATCALC calculates the construction cost of catchbasins required for a given node. The number of catchbasins required at a particular node is based on the direction of flow of storm water to that node.

First, CATCALC assigns the depth index, INDEX, for all the links associated with the current node. This depth index categorizes pipes according to their average depth below ground. The average depth, AVEDEP, is used in an empirical formula to calculate INDEX. If the depth index is less than 1, INDEX is set to 1; if the depth index is greater than 9, INDEX is set to 9. Consequently, INDEX remains between 1 and 9. INDEX is related to depth as shown in Table 1.

Table 1

INDEX	Depth to Invert of Pipe (m)
1	0.0 - 3.0
2	3.0 - 3.5
3	3.5 - 4.0
4	4.0 - 4.5
5	4.5 - 5.0
6	5.0 - 5.5
7	5.5 - 6.0
8	6.0 - 6.5
9	Out of bounds

Second, CATCALC calculates the catchbasin construction cost, CBCOS, if two catchbasins are required for the current node (BASIN=2). The construction cost calculations are based on the CBDPTH, COST16-18 and INDEX. The cost is assigned to the drainage link attached to CNODE. The number of catchbasins associated with the drainage link, LBASIN, is incremented by 2.

Third, CATCALC calculates CBCOS when 3 catchbasins are required at a node (BASIN=3). This cost is based upon CBLEAD, as well as CBDPTH and INDEX. The shortest lead length is determined by comparing CBLEAD(J,1) and CBLEAD(J,2), the catchbasin lead lengths for each link attached to the current node. The smaller of the two lead lengths is used in the cost calculation.

As part of the cost calculation for 3 catchbasins, CATCALC checks if the number of catchbasins for the current node, BASIN, equals the number of drainage links, NFLOW. If not, there is a catchbasin that has not been placed on a

link. This catchbasin is placed on the link with the shortest lead length. To determine which link has the shortest lead length, the two lead lengths for each link, CBLEAD(J,1) and CBLEAD(J,2), are compared to determine the larger of the two. This larger lead length is then compared to CLEAD. If CLEAD is larger, the additional catchbasin is placed on the current link. The number of catch basins for the current link, LBASIN, is then incremented and the catchbasin cost for the current node, LCBCOS, is calculated.

Fourth, CATCALC calculates the construction cost when 4 catch-basins are required at a node (BASIN=4). Catchbasin construction costs for the current node, CBCOS, as well as for the current link, LCBCOS, are calculated based upon the values of CBDPTH, COST16-18 and INDEX. The link construction cost is assigned to the drainage link.

As part of the calculation for 4 catch-basins, CATCALC finds the cost for the last catchbasin and assigns this cost to the current node as well as to the link with the shortest lead length. This link is found in the same manner as used for 3 catch-basins (ie. the larger of the two lead lengths associated with each link is compared with CLEAD and if CLEAD is greater, the current link has the shortest lead length). The construction cost for this additional catchbasin is added to CBCOS and LCBCOS. Control is then returned to the calling program.

5.4.7 CATCALC2 - A Routine to Calculate the Catchbasin Costs for a Link

The variables used by CATCALC2 are:

* IND - temporary depth index value.
* AVEDEP - the average storm sewer depth for the current link (m).
* CLINK - current link.
* LCBCOS - cost of the catchbasins for the current link.
* COST16-18 - elemental unit costs.
* INDEX - depth index for catchbasin.
* CLEAD - length of the catchbasin lead (m).
* CDPTH - depth of the catchbasin from ground level (m).

CATCALC2 finds the catchbasin cost for the link, ignoring the catchbasins at the ends of the links.

First, CATCALC2 finds the depth index, IND, of the link using the method described in the documentation for CATCALC.

Second, CATCALC2 calculates the total catchbasin cost for the given link, LCBCOS. This cost applies only to the catchbasins between intersection on the given link. Control is then returned to the calling program.

5.4.8 CATDEPTH - A Routine to Calculate Catchbasin Lead Length

The variables used by CATDEPTH are:

* NODE1 - origin node of the current link.
* NODE2 - destination node for the current link.
* CLINK - current link in memory.
* CNODE - current node in memory.
* PNODE - previous node in memory.
* CWD - corresponding road width for the current link (m).
* CORWD - corresponding road width (m).
* FROM - identifies the calling routine.
* DIST1 - distance from the current node to the catchbasin (m).
* DIST2 - distance from the current node to the catchbasin (m).
* STLEN1 - total street length; summing the spacing of the catchbasins (m).
* STLEN2 - total street length; the road length of the link (m).
* DSTLEN - difference between STLEN1 and STLEN2 (m).
* LLINK - link length (m).
* ATTLIN - the links associated with the given node.
* SSELE1 - storm sewer elevation of the current link at the current node (m).

* SSELE2 - storm sewer elevation of the current link at the previous node (m).
* SSLINK - storm sewer elevation of the current link.
* CLEAD - length of the catchbasin lead (m).
* CDPTH - depth of the catchbasin from ground level (m).
* BASBED - height of the bedding (m).
* ELEV1 - the difference between the node and storm sewer elevations for the current node (m).
* ELEV2 - the difference between the node and storm sewer elevations for the previous node (m).
* ELEV - the elevation of a node (m).
* AVEDEP - the average storm sewer depth for the current link (m).
* ANG - the slope angle of the storm sewer pipe (rad).
* DPTH - distance from sewer pipe to the catchbasin (m).
* WROAD - width of road of a given link (m).
* OFFSET - the distance from the center of the street to the catchbasin (m).
* BASHGT - height between the link lead and the bedding (m).

CATDEPTH is used to calculate the catchbasin lead length and the distance from the basin lead to ground level.

First, CATDEPTH finds the previous node number, PNODE, by comparing the origin node, NODE1, with the current node, CNODE. If they are equal, the destination node, NODE2, is assigned to PNODE. Otherwise, NODE1 is assigned to PNODE. CATDEPTH then assigns the road width, for the current link and node (either destination node or origin node), CWD, the value CORWD

Second, CATDEPTH identifies whether the node section CATBAS, or the link section CATLIN, calls CATDEPTH. If the CATBAS calls the routine, the distances from the current and previous nodes to the catchbasin, DIST1 and DIST2, are calculated. DIST1 equals half of CWD. DIST2 equals the length of the current link less DIST1. If CATLIN calls the routine, DIST1 equals the length of the link less DSTLEN. DIST2 equals DSTLEN.

Third, CATDEPTH calculates the slope angle of the storm sewer pipe. CATDEPTH finds the storm sewer elevation at both the current and previous nodes, SSELE1 and SSELE2 respectively. CATDEPTH then checks if SSELE1 or SSELE2 are greater than 9000. If so, no lead length calculations are required, since the current node must connect to the sewer system outside the subdivision. CLEAD and CDPTH are assigned to null values and control is returned to the calling program.

If both SSELE1 and SSELE2 are less than 9000, CATDEPTH does lead length calculations. The difference between the

node and storm sewer elevations for current and previous nodes, ELEV1 and ELEV2, are calculated.(See Figures 5 and 6 for an explanation of the calculations for CLEAD.) CLEAD and CDPTH are calculated differently, depending on whether CNODE or PNODE is at a higher elevation. Control is then returned to the calling program.

FIGURE 5 - VARIABLES USED IN CATDEPTH ROUTINE

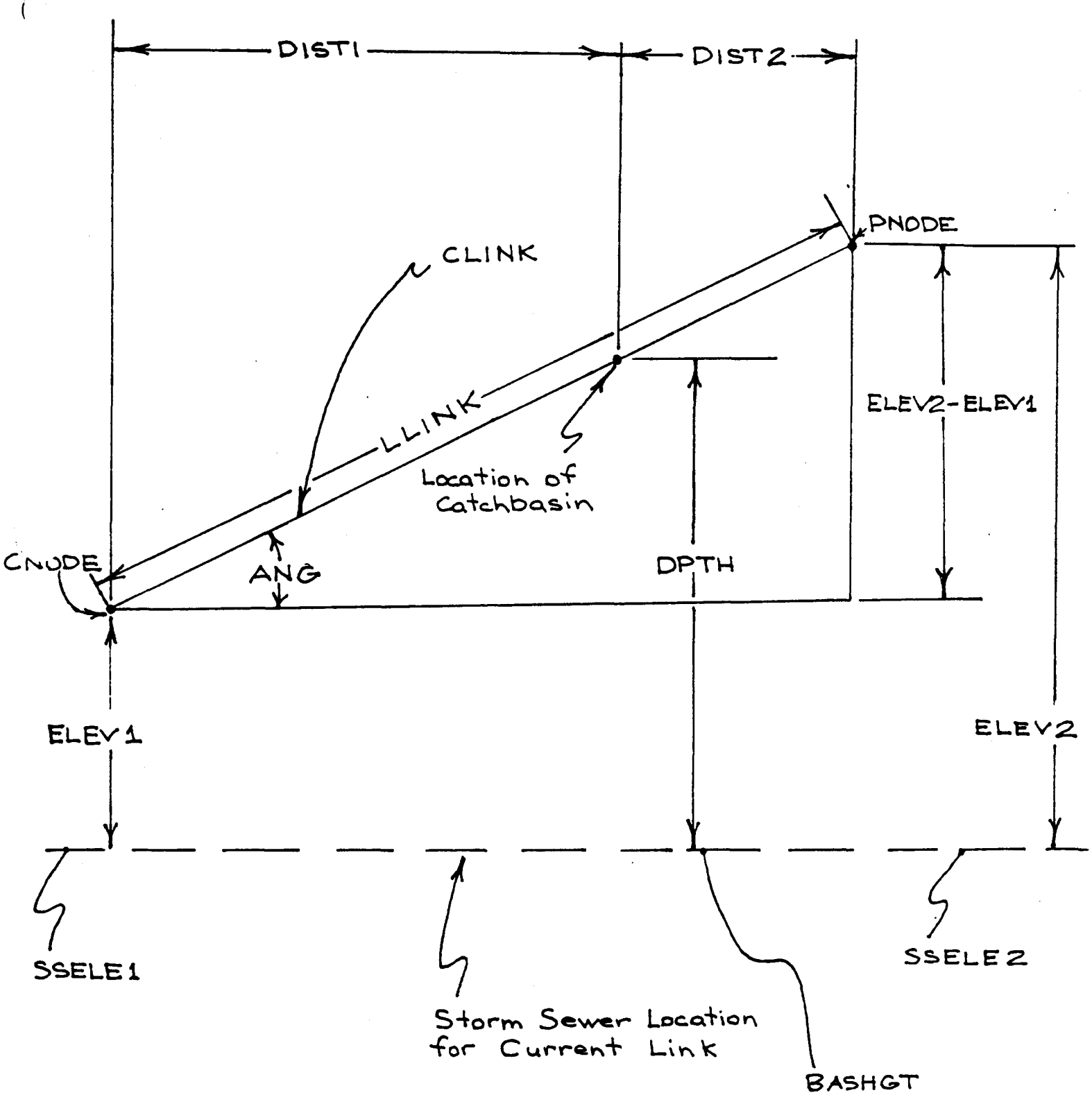
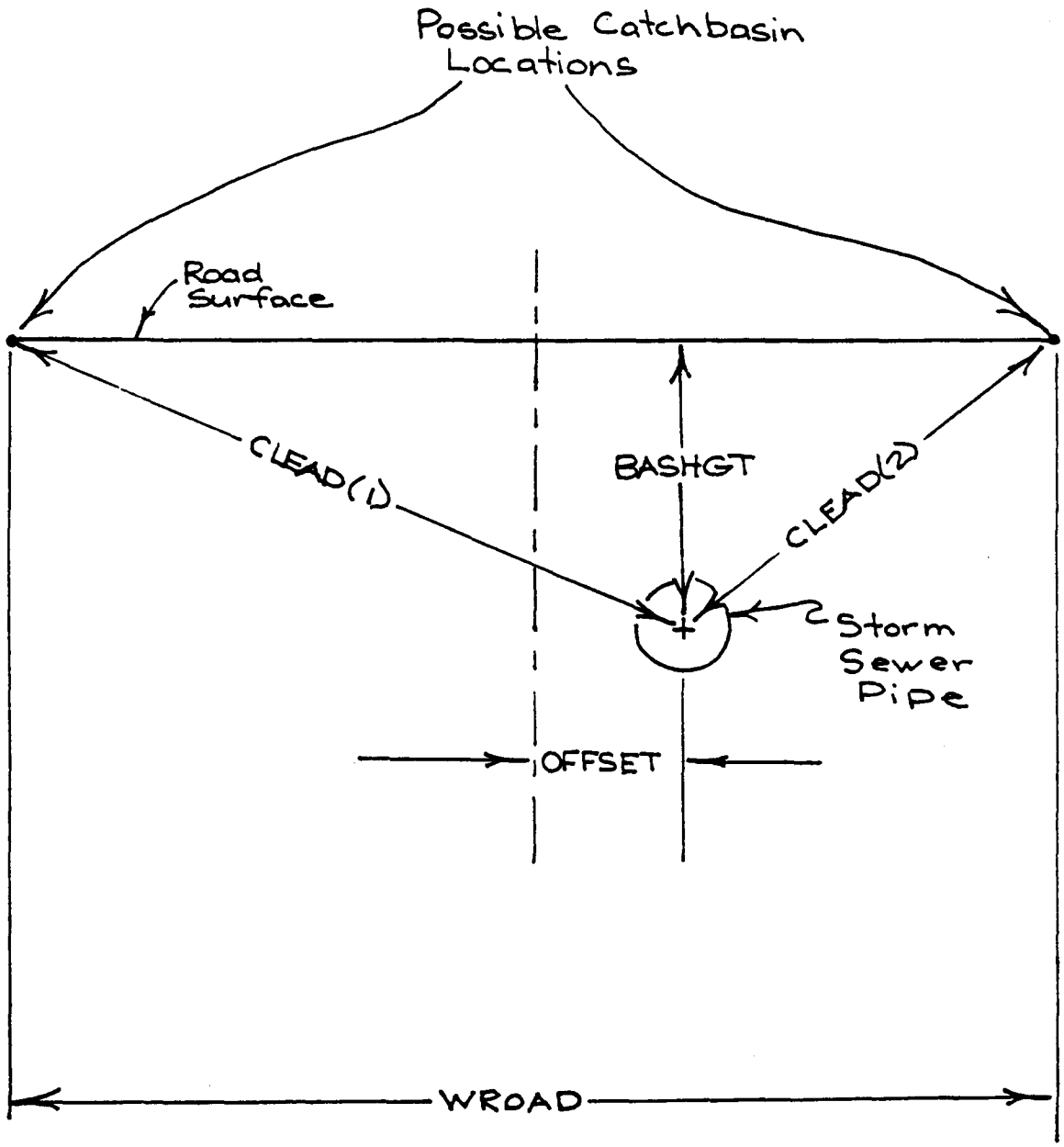


FIGURE 6 - VARIABLES USED IN CATDEPTH ROUTINE



5.4.9 CATLIN - A Routine to Calculate the Number of Catchbasins per Link

The variables used by CATLIN are:

* STLENG - length of a section of street between intersections, intersections and dead ends, intersections and cul-de-sacs (m).
* STRLEN - length of a section of street between two nodes (m).
* ORIGIN - the origin node of the sequence of nodes comprising a street. (One street may be composed of several links.)
* DEST - the destination of the sequence of nodes comprising a street. (one street may be composed of several links.)
* BRANCH - one of a number of links attached to the node.
* LENGTH - length of a section of street (m).
* NBASIN - number of catchbasins per street length (m).
* STRCAT - number of catchbasins per street length. A temporary value.
* SPACAT - spacing of catchbasins (m).
* START - starting link.
* LBASIN - the number of catchbasins required for the current link.

* CLINK - current link in memory.
* DSTLEN - difference in street length (for calculating link catchbasins only; the difference between STLEN1 and STLEN2.) (m).
* STLEN1 - total street length including spacing of catchbasin (m).
* STLEN2 - total street length including the road length of the link (m).
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.
* ATTLIN - links associated with the specified node.
* JOINT - the node connecting two given links.
* NTYPE - node type.
* K,L - counters.
* PATH - links between two intersections, intersections and dead end, intersections and cul-de-sacs.
* SLEN - street length of a given link. (returned from CHECKLEN routine.) (m).

CATLIN calculates the number of catchbasins for a specified street.

CATLIN first determine the entire street length by assigning the value of LENGTH to STLENG. NBASIN is calculated by taking the integer value of:

[(STLENG/SPACAT)-0.5]*2

STRCAT is assigned the value of NBASIN, and accumulates the catchbasins assigned to the street length from CORNER and as the street's links are processed by CATLIN. Each link associated with the street is assigned two catchbasins conditional upon the street length being longer than SPACAT and NBASIN being greater than 0.

Next CATLIN finds the connecting nodes between links that make up a street. Given the conditions above, JOINT is the node that is a part of the street as defined in PATH. CATLIN determines JOINT by comparing: each node connected to each link, and the attached link, with the PATH. JOINT is the CNODE, either NODE1 or NODE2, attached to one of the four ATTLIN that are a part of PATH. If NTYPE of the node being processed is 2 then the node is automatically equal to JOINT.

CATDEPTH and CATCALC2 are called to determine the actual catchbasin lead length and costs at the JOINT or node. CATLIN then increments STLEN1 by the value of SPACAT. The do-loop is exited when each link that is a part of the street has been processed by CATLIN and STLEN1 is greater than STLEN2.

CHECKLEN is called to calculate the value of STLEN2 when STLEN1 is greater than STLEN2 and not all the links have been processed ($K \geq END$). Control is then passed to the calling routine.

5.4.10 CATNODE - A Routine to Calculate the Cost of Catchbasins

The variables used by CATNODE are:

* CNODE - current node.
* CLINK - current link.
* ATTLIN - links associated with the specified node.
* WROAD - width of road of a given link (m).
* SS - array subscript.
* SSELE1 - storm sewer elevation of the current link at the current node (m).
* SSLINK - storm sewer elevation of the current link (m).
* ELEV - the elevation of a node (m).
* BASBED - height of the bedding (m).
* CLEAD - length of the catchbasin lead (m).
* CDPTH - depth of the catchbasin from ground level (m).
* BASHGT - height between the link lead and the bedding (m).
* IND - temporary depth index value.
* INDEX - depth index for catchbasin.
* CBCOS - cost of catchbasins for the node.
* COST16-18 - elemental unit costs.
* BASIN - number of catchbasins for the node.

* LCBCOS - cost of the catchbasins for the current link.
* LBASIN - the number of catchbasins required for the current link.
* NTYPE - node type.
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.
* M1 - type 1 node of the cul-de-sac.
* M2 - type 2 node of the cul-de-sac.
* LINK1 - the link between type 2 node and the other nodes.

The CATNODE routine calculates the cost and number of catchbasins required for cul-de-sacs and dead-ends.

CATNODE calculates the cost of the catchbasin associated with the type 1 and type 2 nodes.

The logic for type 2 nodes is as follows: the link with the highest storm sewer elevation is assigned catchbasin costs. For a type 1 node the link attached to the dead end is assigned catchbasin costs. This link is identified, the catchbasin is assigned and costed (the cost includes that of the lead).

After identifying the links to which costs will be assigned, CDPTH and CLEAD are calculated:

CDPTH = ELEV - SSLINK + BASBED

$$CLEAD = ((0.5 * WROAD)^2 + BASHGT^2)^{1/2}$$

The depth index is also found in order to properly calculate costs for the depth of the catchbasin.

CATNODE checks each node attached to the CLINK. If either node is type 2 then CULCHK is called to identify and calculate the length of the links leading to a possible cul-de-sac. If there is a cul-de-sac (CUL = 1) then the costs for the catchbasin is assigned to the link entering the cul-de-sac(LINK1). Otherwise the costs are assigned to the CLINK. Control is then passed to the calling routine.

5.4.11 CHECKLEN - A Routine to Calculate the Length of Cul-de-Sacs

The variables used in CHECKLEN are:

* NTYPE - node type.
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.
* CLINK - current link.
* CNODE - current node.
* PNODE - previous node in memory.
* RADIUS - radius of the cul-de-sac or tube sac (m).
* SLEN - street length of a given link. (returned from CHECKLEN routine.) (m).
* LLINK - link length (m).
* CORWD - corresponding road width (m).
* LROAD - length of road on a given link (m).

CHECKLEN is used to find the street length, SLEN, of a cul-de-sac/tube. It is necessary that this length be calculated separately from the road length, LROAD, since these values are different for cul-de-sacs and tube sacs.

First, CHECKLEN checks the node type, NTYPE, of the terminal (end) nodes for the link in question. If neither of these nodes are type 1 (cul-de-sac or tube sac), there is no cul-de-sac or tube sac associated with the link. Therefore, SLEN is equal to LROAD for the current link.

If either of the terminal nodes are type 1, however,SLEN must be calculated considering the radius, RADIUS, associated with the node. To do this, CHECKLEN assigns the type 1 terminal node for the current link to CNODE, and the other terminal node to PNODE. If both terminal nodes are type 1, then the origin node for the current link, NODE1, is assigned to CNODE. CHECKLEN then calculates SLEN for the link. If the RADIUS associated with CNODE is zero, and PNODE is type 2 SLEN of the CLINK equals:

SLEN = LLINK - [CORWD(CLINK,1) + CORWD(CLINK,2)] / 2 +RADIUS

If the RADIUS does not equal 0 and NTYPE equals 2 then:

SLEN = LLINK + RADIUS

If the RADIUS equals 0 or the nodes attached to CLINK are not of type 1 then SLEN equals LROAD. Control is then returned to the calling program.

5.4.12 CLINE - A Routine to Create the Border Character Strings Used in the Link Description Tables

The variables used by CLINE are:

* LINE - an variable for line characters.
* LINE1-4 - used to define borders and titles for link description tables.
* MCOL - maximum number of columns in the link description tables.
* XSPA - number of columns in each section of the link description tables.
* XSEC - number of sections in the link description tables.
* XTOTAL - used to format link description table subheadings.
* COL - starting column for link description table.
* CON1 - condition flag indicating the shift of a table section.
* CON2 - condition flag indicating the shift of a table section.
* I - loop index variable.

CLINE creates the character strings that form the borders for the link description tables. These link description tables are output in the routine ANALYSIS.

First, CLINE clears the character variables, LINE and LINE1 to LINE4. The variables are cleared by assigning all the characters within them to blanks.

Second, CLINE assigns the characters in these variables to either dashes ("-"), exclamation points ("!"),"plus" signs ("+") or spaces using a section of code for each variable. The dashes in the variables serve as the table borders, while the exclamation points and plus signs indicate the table sections.

```
eg. :     ...----------------+----------------...
                             !
             Table Section   !   Table Section
                             !
          ...----------------+----------------...
```

For each variable, CLINE uses a loop to assign the main characters. Following the loop, additional statements are used to assign the remainder of the characters for each variable. XTOTAL is used to determine where the "!" and "+" characters (the table section indicators) should appear in the variables.

5.4.13 COR - A Routine to Calculate the Pavement Area for Straight Roads Streets and Bulb Curves

The variables used by COR are:

* ASAC - area of the road connected to a bulb curve (m^2).
* LEN - road length connected to a bulb curve (m).
* ACIR - area of the bulb portion of the bulb curve (m^2).
* AEXC - area of quarter circle minus area of triangle (m^2).
* CNODE - current node in memory.
* PNODE - previous node in memory.
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.
* RADIUS - radius of the cul-de-sac/tube associated with a node (m).
* RAD - half the radius associated with the current node (m).
* S1 - half the perimeter of a triangle (m). (Used in Heron's Formula.)
* WROAD - width of road of a given link (m).
* ATRI - area of triangle calculated using Heron's Formula (m^2). [Heron's formula:

$$[S1 * (S1 - WROAD) * (S1 - RAD) * (S1 - RAD)]^{1/2}$$

* H - height of triangle (m).
* ANG - apex angle of the triangle (rad).
* ASEC - area of the sector of the circle (m^2).
* AEXC - area of the segment of the circle (m).
* ROAD - length of the straight part of the curve (m).
* LLINK - length of the current link (m).
* AEND - area of the straight part of the curve (m^2).
* ACUB - area of the curb (m^2).
* CURB - radius of the curb (m).
* ASAC - area of the road, including bulb area, associated with the current link (m^2).
* BULB - number of bulb curves attached to the link.

COR is used to calculate the pavement area for a node and attached links associated with a bulb curve. In order to calculate the bulb area, COR assumes that it can be modeled as the sector of a circle.(See figure 7)

First, COR assigns the bulb node to the current node, CNODE. It then calculates AEND by multiplying WROAD by ROAD. Half of the bulb curve pavement area is assigned to a link. ACIR is calculated as half of the area of a circle, ASEC is a quarter circle, and ATRI is the triangle area. ACUB is calculated as 1/8 of a circle whose radius is measured from the inside of the sidewalk area. This area must be

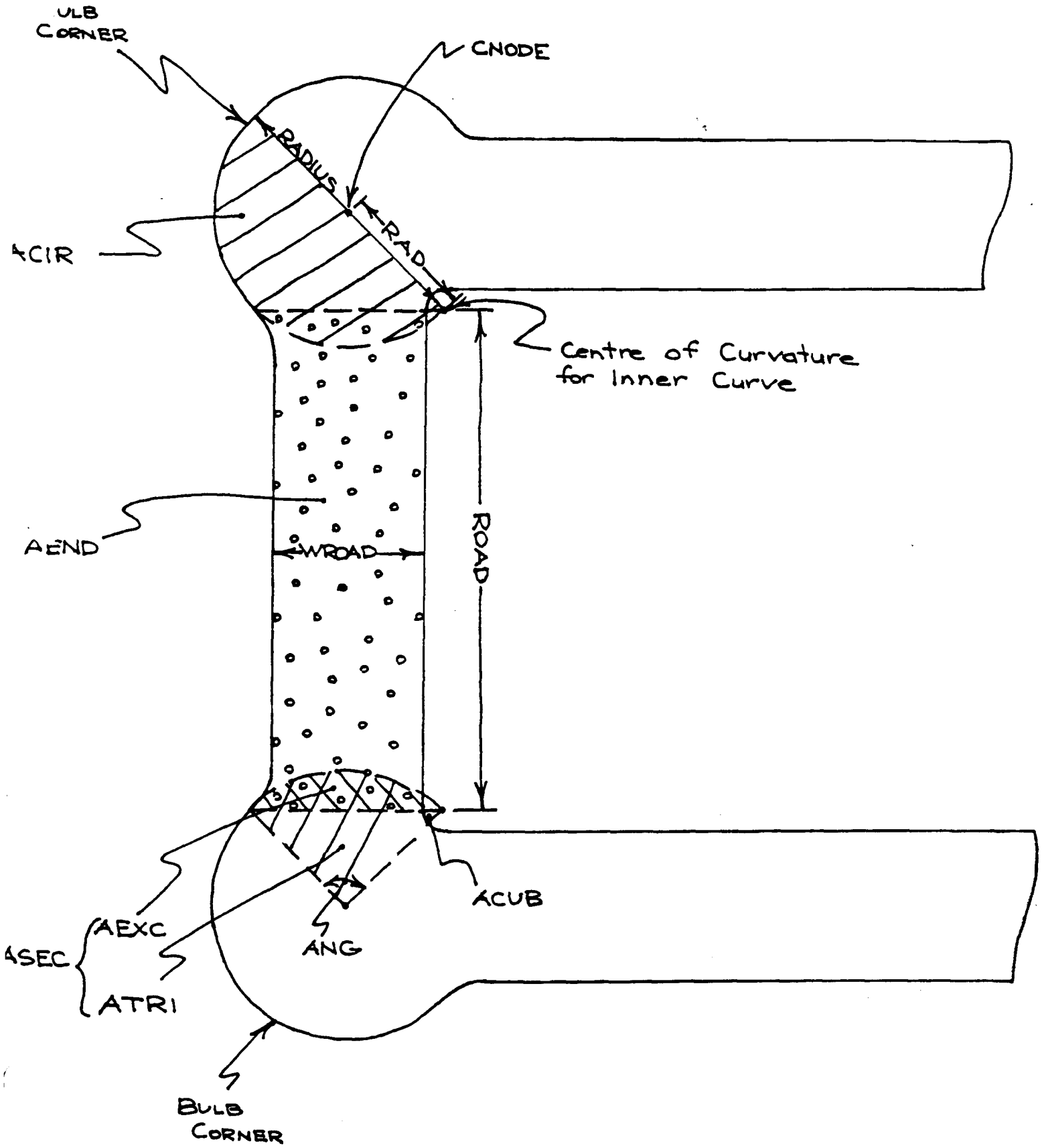
separately calculated because it is part of the excess area calculated in ACIR. ASAC is finally calculated as being:

ASAC = AEND + ACIR - AEXC - ACUB

If two bulb curves are attached to the link, ASAC equals:

ASAC = AEND+ACIR(1)+ACIR(2)-AEXC(1)-AEXC(2)-BULB*ACUB

FIGURE 7 - VARIABLES USED IN COR ROUTINE



5.4.14 CORCHK - A Routine to Identify Bulb Corners

The variables used by CORCHK are:

* BULB - flag indicating the presence of a bulb curve.
* NBL - the number of links between two intersections, or intersection and dead-end, or intersection and cul-de-sac.
* CLINK - current link.
* PATH - links between two intersections or intersection and dead end, or intersection and cul-de-sac.
* ORIGIN - origin node of the trace.
* BRANCH - branch of the origin node.
* NODE1, N1 - origin node for the current link.
* NODE2, N2 - destination node for the current link.
* NTYPE - node type.
* RADIUS - radius of the cul-de-sac or tube sac associated with the current node (m).

CORCHK is used to find the number of corners that have bulbs. It checks all the links between the ORIGIN and DEST nodes that are a part of the BRANCH. From CLINK and NTYPE information it determines whether a node has a bulb associated with it, NTYPE equals 2. The check is made of the node to determine whether or not is has a radius associated with it. If so, the flag BULB is incremented to indicate a

bulb presence. Control is then passed back to the calling routine.

5.4.15 CORNER - is a Routine to Calculate the Number of Catchbasins on Bulb Corners

The variables used by CORNER are:

* LENGTH - the length of a section of a street (m).
* NBL - the number of links between two intersections, or intersection and dead-end, or intersection and cul-de-sac.
* START - the starting link.
* CLINK - current link.
* ORIGIN - the origin node of the sequence of nodes comprising a street. (One street may be composed of several links.)
* BRANCH - the branch of the path.
* NTYPE - node type.
* NODE1,N1 - origin node for the current link.
* NODE2,N2 - destination node for the current link.
* CNODE - current node.
* RADIUS - radius of the cul-de-sac or tube sac (m).
* DEST - the destination node of a sequence of nodes comprising a street.
* SLEN - street length of a given link (m). (Returned from CHECKLEN routine.)
* STRCAT - the number of catchbasins per street length.

CORNER is used to calculate the number of catchbasins on bulb corners. The routine first identifies the links between intersections and dead-ends or cul-de-sacs and finds bulb corners.

CORNER then finds the origin and destination nodes for the current link. It checks the radius of the node. If RADIUS is greater than 0 then CATNODE is called to calculate the catchbasin costs and CATLIN is called to calculate the number of catchbasins to put on the link. If RADIUS equals 0 CHECKLEN is called to calculate the LENGTH of the street. Control is then passed to the calling program.

5.4.16 COUNTER - A Routine to Count the Number of 3-Way and 4 Way Intersections, Cul-de-Sacs, and Exit Nodes in the Subdivision

The variables used by COUNTER are:

* NCUL - number of cul-de-sacs in the subdivision.
* N3WAY - number of 3 way intersections in the subdivision.
* N4WAY - number of 4 way intersections in the subdivision.
* NEXIT - number of exits in the subdivision.
* NDEAD - number of dead ends in the subdivision.
* NNODE - number of nodes in the subdivision.
* NTYPE - node type.
* RADIUS - radius of the cul-de-sac or tube sac (m).

The routine checks NTYPE to determine whether the node is a : dead-end
cul-de-sac
3-way intersection
4-way intersection
exit

and increments the appropriate counters.

5.4.17 CULCHK - A Routine to Identify and Calculate the Length of Links Leading to a Cul-De-Sac

The variables in CULCHK are:

* CUL - cul-de-sac type 2 node flag.
* LEN - total length of links (m).
* CLINK - current link.
* NTYPE - node type.
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.
* N1 - node type for other node in cul-de-sac.
* N2 - type 1 node for cul-de-sac.
* CNODE - current node.
* PNODE - previous node in memory.
* LINK1,2 - the link terminating in or entering into the cul-de-sac.
* ATTLIN - links associated with the specified node.
* WROAD - width of road of a given link (m).
* RADIUS - radius of the cul-de-sac/tube (m).
* LLINK - link length (m).

CULCHK is used to calculate the link length ending in a cul-de-sac. It identifies the entrance or end nodes of the cul-de-sac as PNODE and CNODE. CNODE is always type 2.

Next it identifies and assigns the links entering the cul-de-sac as LINK2. LINK2 is between a node of type 1 and a

node of type 2. This node is identified by checking the CLINK's attached to the CNODE to see that the are not easements (ie. CLINK = 0 or WROAD = 0), and NTYPE of the PNODE equals 1. PNODE being the node attached to CLINK that is not of type 2. LINK1 is assigned to be the link that does not have a PNODE NTYPE equal to 1.

Finally, the total length of the links are added:

LEN = LLINK(LINK1) + LLINK(LINK2)

and CUL equals 1 when both LINK1 and LINK2 do not equal 0 and RADIUS has a value. Control is then returned to the calling program.

5.4.18 DEPTH - A Routine to Determine the Depth Index of a Given Pipe

The variables used by DEPTH are:

* E1 - elevation of origin node (NODE1) (m).
* E2 - elevation of destination node (NODE2) (m).
* PE1 - elevation of pipe at the origin node (m).
* PE2 - elevation of pipe at the destination node (m).
* D1,D2 - elevation of the origin and destination nodes minus the elevation of the pipes (m).
* DPTH - average depth into ground of pipe (m).
* DI - depth index (1-9).

DEPTH calculates the depth index (DI) of specific pipes thus categorizing pipes according to their average depth below ground. First, DEPTH calculates the depth below ground for each node belonging to the specified pipe. Then the average depth of the pipe is determined. The average depth is used in an empirical equation to calculate DI. If DI is less than 1, the index is set to 1; if DI is greater than 8, the index is set to 9. Consequently DI remains between 1 and 9. DI is related to depth as shown in Table 2.

Table 2

DI	Depth to Invert of Pipe (m)
1	0.0 - 3.0
2	3.0 - 3.5
3	3.5 - 4.0
4	4.0 - 4.5
5	4.5 - 5.0
6	5.0 - 5.5
7	5.5 - 6.0
8	6.0 - 6.5
9	Out of bounds

5.4.19 DIAMTR - A Routine to Determine the Pipe Slope Required to Meet or Exceed the Minimum Acceptable Water Velocity in a Pipe

The variables used by DIAMTR are:

* D - diameter of pipe (m).
* Q - required flow in pipe (cms).
* M - slope of pipe.
* N - Manning's roughness coefficient.
* VMIN - minimum acceptable water velocity in the pipe (m/s).
* HDEPTH - over-estimated value of depth of water in pipe (m).
* LDEPTH - under-estimated value of depth of water in pipe (m).
* DEPTH - estimated value of depth of water in pipe (average of HDEPTH and LDEPTH) (m).
* PRCNT - percentage that the depth is of the diameter.
* WP - wetted perimeter (m).
* AREA - cross-sectional area of water in the pipe (m^2).
* R - hydraulic radius (m).
* QEST - an estimate of the flow at the estimated depth using Manning's equation (cms).
* DIFF - the difference between the estimated flow and the required flow (cms).
* V - the velocity of water in the pipe (m/s).

* VMAX - maximum acceptable water velocity in the pipe (m/s).

DIAMTR calculates the pipe slope required to ensure sufficient water velocity within the pipe. First, the depth of water in the pipe is calculated using the bisection method. HDEPTH is set to the diameter of the pipe and LDEPTH is set to zero. The average of HDEPTH and LDEPTH becomes the estimated depth. This depth is used to estimate the corresponding flow. If the estimated flow is less than the actual flow and if the flows do not fall within defined tolerances, LDEPTH is set to Depth. If the estimated flow is greater than the actual flow and if the flows do not fall within the defined tolerances, HDEPTH is set to depth. Then a test is made to ensure that HDEPTH and LDEPTH converge. If they do converge, the average of HDEPTH and LDEPTH becomes the new estimated Depth and the process is repeated. If they do not converge, control returns to the calling routine.

If the estimated flow and the actual flow fall within the defined tolerances then the depth of water in the pipe has been found. The water velocity is consequently calculated for the current slope. If the velocity is less than the minimum allowable velocity, the slope is recalculated using VMIN, if the velocity is greater than VMAX the slope is recalculated using VMAX. In either case, the new slope is used to calculate a new depth of water in

the pipe and a new velocity. On the other hand, if the velocity is within the defined tolerances, the correct pipe slope has been found and control returns to the calling program.

5.4.20 DMAKER - A Routine to Allocate Subdivision Elements to Collector Streets or Links at Major Intersections

The variables used by DMAKER are:

* LINK - stores the collector/major local link.
* NCOL - number of collector streets.
* CLINK - current link.
* ATTLIN - links associated with the specified node.
* WROAD - width of road of a given link (m).
* TROAD - type of road or street.
* QUAN - number of signs/manholes/lights at the node.
* NODE1,N1 - origin node for the current link.
* NODE2,N2 - destination node for the current link.
* EXIT - exit node.
* M1 - node type for other node in cul-de-sac.
* M2 - type 1 node for cul-de-sac.
* M3 - link between type 2 node and other type node.
* CUL - flag indicating a cul-de-sac.

Initially DMAKER sets the collector or major links counters, (LINK), to zero. It checks the TROAD of each link attached to the CNODE, and if it is a collector(ie. TROAD = 1), identifies and counts these links in NCOL.

Next a loop is set up to assign the particular element being passed from the calling program to the link on which it exists.

The loop first checks NCOL to determine how many collector streets occur at the CNODE. If NCOL equals 0, DMAKER checks each ATTLIN to ensure that they are not easements. If they are they are not allocated any subdivision elements. Each of the ATTLIN are checked to see if their nodes are NTYPE 2. If they are, CULCHK is called to identify the links leading to a cul-de-sac. If CUL equals 0 and there is no cul-de sac and CLINK is not equal to LINK1 then LINK(N) is assigned to the N'th LINK depending upon which element the do-loop is processing (how many of the QUAN times the loop has been executed). If LINK1 equals CLINK or CUL is not 0; or if the ATTLIN are not equal to 2, then LINK(3) equals CLINK.

If NCOL equals 1 and QUAN is greater than 1 then DMAKER checks the ATTLIN of CNODE to see if they are easements and TROAD equals 1. If these conditions hold then LINK(1) is assigned the value of CLINK to represent allocation of the subdivision element to the major LINK(1).

If NCOL is greater than or equal to 2 then each subdivision element is assigned to LINK(1) as long as : each ATTLIN to the CNODE is not and easement, TROAD of each ATTLIN equals 1, and N1 and N2 are not exit nodes

(EXIT(N1)=0). If they are exit nodes then LINK(3) equals CLINK.

If NCOL is greater than 0 and the LINK(1) is assigned, LINK(2,3) must also be assigned. A nested loop checks ATTLIN, each of the four possible links attached to the CNODE, to see if they are easements. DMAKER then checks for ATTLIN NTYPE of 2 and calls CULCHK. If CUL equals 0 then LINK(2) equals CLINK, if CUL is greater than 0 LINK(3) equals CLINK.

After DMAKER has completed execution of the initial do-loops it checks its work to ensure that a link has been identified for the allocation of subdivision elements. If QUAN equals 1 and LINK(1) equals 0 this indicates non-existence of a major link. The earlier do-loop identified the CLINK as LINK(3). LINK(1) is now assigned the value of LINK(3).

LINK(2) is assigned the value of LINK(3) when LINK(2) equals 0, LINK(1) is not equal to 0 and QUAN equals 2. This ensures that LINK(2) has a value after the initial do-loops have completed their processing.

If LINK(1,2) both equal 0 and QUAN equals 2 then each ATTLIN of CNODE is checked to see if they are easements. If not then LINK(1,2) are assigned an ATTLIN.

If the NTYPE equals 5 (an 'other' type node) , then each ATTLIN of CNODE not equal to 0 is assigned to LINK(1). Control is then passed to the calling routine.

5.4.21 DMAKER2 - A Routine to Assign Subdivision Elements to the Local Street Links

The variables used by DMAKER2 are:

* LINK - the local link.
* YIELD - the flag indicating yield signs at the node/link.
* STOP - the flag indicating stop signs at the node/link.
* QUAN - the number of signs at the node/link.
* CLINK - current link.
* ATTLIN - links associated with the specified node.
* CNODE - current node.
* WROAD - width of road of a given link (m).
* TROAD - type of road/street.

The DMAKER2 routine assigns street signs to the local streets. First the routine determines what the calling program requires in terms of elements (ie. yield or stop signs). Then DMAKER2 assigns these elements to the local links.

If YIELD is greater than 1 each sign at the CNODE is allocated to the LINK's(1,2,3), as long as they are ATTLIN of CNODE, not easements (WROAD not equal to 0 and CLINK not equal to 0) and TROAD is equal to 1.

If there is 1 YIELD sign and 1 or more STOP signs at the intersection DMAKER2 checks each ATTLIN of the CNODE for an easement. If there are none and TROAD of the ATTLIN equals 2 then LINK(1) equals the ATTLIN. If TROAD does not equal 2 LINK(2) is assigned the value of ATTLIN.

If YIELD equals 1 and STOP equals 0 then DMAKER2 checks each ATTLIN of the CNODE for an easement and TROAD equal to 2. If these conditions are met then DMAKER2 assigns LINK's(1,2,3) to each ATTLIN. Control is then passed to the calling routine.

5.4.22 DRAIN - A Routine to Determine the Flow Direction of Drainage With Respect to the Node

The variables used by DRAIN are:

* FLAG - indicates the direction of the drainage flow. (0 is out of the node; 1 is into the node).
* NODE1 - origin node of the current link.
* NODE2 - destination node of the current link.
* PNODE - previous node in memory.
* CNODE - current node in memory.
* CLINK - current link.
* NTYPE - node type.
* RADIUS - radius of the cul-de-sac or tube sac (m).
* LINK - the link associated with a type 2 node.
* ATTLIN - links associated with the specified node.
* WROAD - width of road of a given link (m).
* ELEV - the elevation of a node (m).

DRAIN is a routine that is called by CATBAS to determine the drainage flow direction to or from a specified node. The routine first sorts the origin and destination nodes to ensure they are identified as PNODE or CNODE. This is done by checking to see if NODE1(CLINK) equals CNODE. If so then PNODE equals NODE2(CLINK). If not PNODE equals NODE1(CLINK).

The routine then checks NTYPE's to identify cul-de-sacs. If they are, then the routine checks the nodes that are at the exit of the cul-de-sac. In the case of a cul-de-sac the node of type 2 at the mouth of the cul-de-sac might have the same elevation as the interior cul-de-sac node. Then the routine must find the next node to determine if it is higher than the cul-de-sac node. This is done by meeting the conditions of CNODE: NTYPE equal to 1, RADIUS not equal to 0, and NTYPE(PNODE) equal to 2. A do-loop is then entered checking each ATTLIN of the CNODE for an easement and ensuring that each CLINK is not the same as the ATTLIN being processed. If these conditions are met and NODE1 of the ATTLIN being processed is the PNODE, then the other node attached to the ATTLIN is the new PNODE and is compare with the elevation of the CNODE in the cul-de-sac. The FLAG is either 1 or 0 depending upon whether the CNODE or PNODE elevation is greater. Then control is returned to the calling program.

5.4.23 DSSDPTH - A Routine to Allocate Domestic Sewer Elevations to Nodes

The variables used by DSSDPTH are:

* ATTLIN - links associated with the specified node.
* CLINK - current link.
* CNODE - current node in memory.
* DSLINK - domestic sewer elevation of a given link (m).
* NODE1 - origin node of the current link.
* NODE2 - destination node for the current link.
* DPTH1 - the depth of excavation at the origin node (m).
* DPTH2 - the depth of excavation at the destination node (m).
* ELEV - ground elevation of the node (m).

The DSSDPTH routine is called by the DMSTC routine to assign the correct elevation and excavation depths to the nodes for domestic sewer pipe. CLINK, DPTH1, DPTH2 are the variables being passed to DSSDPTH.

Each link associated with a certain node is checked to see if it is the CLINK being passed. If so then the elevation is calculated as being the ELEV-DPTH1. An identical calculation is performed for the destination and origin nodes.

5.4.24 FINDPATH - A Routine to Identify the Links Between Intersections, Dead-ends, and Cul-de-sac Nodes

The variables used by FINDPATH are:

* EASE - indicates the presence of an easement attached to a node.
* S - indicates the starting node for the routine.
* N - starting node value passed from calling routine.
* C - the number of links between two intersection,dead-end nodes.
* J - the ending node value of the street without branches.
* L - current link, one of four possible links attached to the starting node.
* ATTLIN - links associated with the specified node.
* WROAD - width of road of a given link (m).
* PATH - links between two intersections or intersection/dead end/cul-de-sac.
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.
* NTYPE - node type.

FINDPATH determines the PATH or the route from one intersection to the next with no branches off of the street between. From the starting node the routine checks each node

and link attached to this node. If a branch of more than one link occurs at the node this signifies the end of the street. If WROAD of ATTLIN equals 0, signifying an easement, J is incremented by 1. If J is 5 or over EASE is reset to 0 and control is returned to the calling routine.

If no easement is encountered then the ATTLIN being processed is stored in PATH(N,J,C)

N is the starting node.

J is the ending node of the PATH.

C is the counter of the number of links in the PATH.

FINDPATH increments S to be the value of the other node attached to the ATTLIN just processed. This nodes NTYPE is checked to see if it is a value other than 2. If so the routine ends. The ATTLIN to this node are checked as to whether or not they are easements if so they are noted as explained in the previous paragraph and control is passed to the calling routine.

The routine is called by LIGHT to calculate the number of links that require lights on the street.

5.4.25 GOGRAD - A Routine to Calculate Present, Future Values and Annuity Amounts

The variables used by GOGRAD are:

* INTRST - interest rate used for discounting present values.
* RATE - INTRST/100.
* GDRATE - growth rate of the capitalized or annuity amount.
* PWORTH - present value of a base amount.
* BASE - base amount of initial cash outlay at time zero.
* W - intermediate variable for present value calculations.
* FWORTH - future value of a present value amount.
* AWORTH - annuity amount equivalent to the PWORTH.
* UNFORM - function used to calculate annuity values given a present value lump sum, time period and interest rate.

GOGRAD evaluates the present and future values of current costs. These amounts indicate the costs of capital for various construction and operating expenses incurred in the subdivision.

The routine determines which formula to use in calculating PWORTH by comparing growth and interest rates.

Next the FWORTH is calculated. The routine calls the function UNFORM to calculate the annuity amounts given a PWORTH.

If GDRATE equals RATE:

PWORTH = BASE * N / (1.0 + GDRATE)

W = (1.0 + RATE) / (1.0 + GDRATE) - 1.0

If GDRATE is greater than RATE:

$PWORTH=BASE/(1.0+GDRATE)*\{[(1.0+W)^{N}-1.0]/W\}$

W = (1.0 + RATE) / (1.0 + GDRATE) - 1.0

$FWORTH = PWORTH * (1.0 + RATE)^{N}$

AWORTH = UNFORM(INTRST,N,PWORTH)

5.4.26 GRAPH - A Routine to Graph Various Cost Results of RPCAM

The variables used by GRAPH are:

* TITLE - graph title.
* XNAME - x-axis title.
* YNAME - y-axis title.
* COL - starting column for graph.
* YSPA - number of spaces in each vertical section.
* XSPA - number of spaces in each horizontal section.
* XSEC - number of horizontal sections.
* YSEC - number of vertical sections.
* INVMIN - minimum investment rate of return.
* INVMAX - maximum investment rate of return.
* REVMIN - minimum revenue.
* REVMAX - maximum revenue.
* TCOS - total costs of subdivision construction.
* MROW - maximum rows on graph.
* MCOL - maximum columns on graph.
* XVAL - revenue per horizontal section.
* XVAL1 - revenue per column.
* YVAL - investment rate of return per horizontal section.
* YVAL1 - investment rate of return per row.
* LENGTH - length of the titles.

* LEN - function that calculates character string lengths.
* FRONT - number of spaces in front of the character string.
* INT - function that converts real numbers to integers.
* LINE - line characters ie. a space.
* XTOTAL - horizontal spacing counter.
* YTOTAL - vertical spacing counter.
* VAL - value to be plotted.
* SIDE - one character.
* SLOPE - slope of the graph.

The GRAPH routine initializes all titles and axis names. The function LEN is used to determine the length of the titles for centering on the graph. Next the vertical and horizontal grids are drawn. Followed by the calculation of and the printing of the grid values. The SLOPE is calculated from a given hardcoded value, SLOPE and output.

5.4.27 GROUP1 - A Routine to Create Watermain Tributary Data

The variables used by GROUP1 are:

* NTYPE - The type of node: cul-de-sac, dead-end, 3-way, etc. If a node is an access/egress node, then NTYPE will initially have a negative value; later, when the access/egress points no longer need to be identified, the negative values are converted to positive values.
* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.
* CHEKIN - keeps track of which links have been tested for connection to an access/egress point.
* REFNOD - the node currently being referenced.
* REFLNK - the link currently being referenced.

* CHONUM - number of choices that were made in designing a particular system.
* CHOLNK - link numbers of links involved in each choice.
  - (index from 1 to CHONUM, 1-3)
  - 1 is the number of the link not chosen.
  - 2 is the number of the link chosen.
  - 3 is the number of the node at which the choice occurred.
* NUMPTH - number of possible path choices a test can follow.
* PATHS - a list of all possible path choices.
* CHECK - indicates which links are in the system being evaluated.
* LIST1-2 - arrays for grouping the data together.
* SRTIDX - index into sort array.
* SORT - an array for sorting the groups of data.
* NUMNOD - number of nodes involved in the system being evaluated.
* NODES - which nodes are in the system being evaluated.
  - (index from 1 to NUMNOD, 1-5)
  - 1 is the identity of the node around which the group is formed.
  - 2-4 are the 'from' or supply links.
  - 5 is the 'to' or receptor link.

* FLAG - flag to indicate when a sort cycle is finished.
* NUMEXT - number of subdivision exits in the system being evaluated.
* EXTREF - array holding indexes into each NODES group which is centered around an exit node.
* NUMLNK - number of links in the system.
* LINKS - links involved in the system.
* TBTRY - tributary data returned to the calling routine.
* L,L1-3 - loop index variables.
* X,X1-7, - miscellaneous variables.

GROUP1 is used to create tributary data for the watermain system. (Tributary data shows the relationship between feeding and receiving links so that the water flow can be traced from origin to destination.)

First, two nested loops are used to identify nodes that are connected to more than one link. The first loop indexes each node in the system. The second loop compares the link descriptors of every link in the system to the indexed node. When a link that is connected to the indexed node is discovered, a counter is incremented by one. Upon completion of the inner loop, if the counter is greater than one, the node number is stored in the first element of NODES and the corresponding element in CHECK is flagged with a one.

Second, the access/egress nodes are identified and included in NODES. A loop indexes links so that NTYPE for each link descriptor can be checked for a negative value. (A negative NTYPE value indicates that the node is an access/egress node.) If NTYPE is negative and the node has not been used before (as indicated by CHECK), it is added to NODES and the corresponding element in CHECK is flagged with a one. Then the number of access/egress nodes are counted. When one is found, the corresponding element number for NODES is stored in EXTREF (exit reference).

Third, the tributary data for each access/egress node are formed. A loop indexes each access/egress node in NODES. PATHS7 is called to identify links adjacent to the access/egress nodes. These links, known as 'from' links, (i.e. a link to which water is flowing) are stored in elements 2, 3, and 4 of NODES. Element 5 is given the value 150 to indicate that the corresponding 'to' link leaves the system. (Here the 'to' link represents the main water supply.) Also, the number of each 'from' link is stored in LIST1.

Fourth, a DO-WHILE loop is used to create the remaining tributary data. Another loop indexes each link in LIST1; these links become the 'to' links for the new group of NODES tributary data. The indexed link in LIST1 is stored in REFLNK for reference. PATHS2 is called to identify the unused links attached to REFLNK. Then the node belonging to

REFLNK that is not attached to a previous tributary data group is found and stored in REFNOD; REFNOD is the node around which the next group of data is formed. A loop is used to find the NODES entry containing REFNOD. REFLNK is stored in element 5 of this NODES entry, and the links identified by PATHS2 are stored in locations 2, 3 and 4. These links are also flagged in CHEKIN and stored in LIST2.

When every link in LIST1 has become part of a new piece of tributary data, LIST1 is cleared and LIST2 is copied into LIST1. The entire process is then repeated with the next set of newly found links. FLAG is set to one when all links have been found and all data groups have been formed (i.e. LIST2 is empty), consequently terminating the DO-WHILE loop.

Fifth, in the process of creating the data, some of the nodes originally included as possible centers of data may have become excluded from the system when various links were grouped with other nodes. Nodes excluded from the system are identified by looking for NODES entries having no 'to' link (i.e. element 5 in NODES equals zero) and deleted. When an entry is deleted, all successive NODES entries are shifted to fill the gap. The elements of NODES are also reorganized so that all 'from' links fill the first elements of NODES (2-4). Therefore, a group with only one 'from' link, will contain a zero in elements 3 and 4, and the 'from' link number in element 2.

Sixth, the tributary data is reorganized so that the 'from' links precede all related 'to' links. The access/egress nodes are re-identified in NODES. CHECK is purged and then the corresponding element number for each access/egress node is stored in LIST1 and flagged in CHECK. For each tributary group represented by the LIST1 entries, the NODES information is copied to SRTIDX. Again, a DO-WHILE loop is used to reorganize the tributary data and a loop indexes each LIST1 group. For each 'from' link in a LIST1 group, a related and unused 'to' link group is found in NODES and copies to SRTIDX; its index is stored in LIST2, then another LIST1 group is indexed. When all groups in LIST1 have been used, LIST2 is copied into LIST1 and the process is repeated.

When all the groups have been ordered, LIST2 will be empty, FLAG will be set to one, and the DO-WHILE loop will terminate.

Since SRTIDX is now organized so that the access/egress point are near the beginning of the list and all successive groups are near the end (i.e. opposite to what was needed), SRTIDX is copied in reverse order to TBTRY. However, only the 'from' and 'to' links are copied since the node numbers are not required by succeeding routines. Control then passes to the calling routine.

5.4.28 GROUP2 - A Routine to Create Domestic and Storm Sewer Tributary Data

The variables used by GROUP2 are:

* NTYPE - The type of node: cul-de-sac, dead-end, 3-way, etc. If a node is an access/egress node, then NTYPE will initially have a negative value; later, when the access/egress points no longer need to be identified, the negative values are converted to positive values.
* NODE1 - an array which holds the node of highest elevation for each link.
* NODE2 - an array which holds the node of lowest elevation for each link.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being test, and then moves along all possible paths in an attempt to reach an access/egress point.
* CHEKIN - keeps track of which links have been tested for connection to an access/egress point.
* REFNOD - the node currently being referenced.
* REFLNK - the link currently being referenced.

* CHONUM - number of choices that were made in designing a particular system.
* CHOLNK - link numbers of links involved in each choice.
  - (index from 1 to CHONUM, 1-3)
  - 1 is the number of the link not chosen.
  - 2 is the number of the link chosen.
  - 3 is the number of the node at which the choice occurred.
* NUMPTH - number of possible path choices a test can follow.
* PATHS - a list of all possible path choices.
* CHECK - indicates which links are in the system being evaluated.
* LIST1-2 - arrays for grouping the data together.
* SRTIDX - index into sort array.
* SORT - an array for sorting the groups of data.
* NUMNOD - number of nodes involved in the system being evaluated.
* NODES - which nodes are in the system being evaluated.
  - (index from 1 to NUMNOD, 1-5)
  - 1 is the identity of the node around which the group is formed.
  - 2-4 are the 'from' or supply links.
  - 5 is the 'to' or receptor link.

* FLAG - flag to indicate when a sort cycle is finished.
* NUMEXT - number of subdivision exits in the system being evaluated.
* EXTREF - array holding indexes into each NODES group which is centered around an exit node.
* NUMLNK - number of links in the system.
* LINKS - links involved in the system.
* TBTRY - tributary data returned to the calling routine.
* L,L1-3 - loop index variables.
* X,X1-X7, - miscellaneous variables.

GROUP2 is used to create domestic and storm sewer tributary data. (Tributary data show the relationship between feeding and receiving links so that the water flow can be traced from origin to destination.)

First, two nested loops are used to identify nodes that are connected to more than one link. The first loop indexes each node in the system. The second loop compares the link descriptors of every link in the system to the indexed node. When a link that is connected to the indexed node is discovered, a counter is incremented by one. Upon completion of the inner loop, if the counter is greater than one, the node number is stored in the first element of NODES and the corresponding element in CHECK is flagged with a one.

Second, the access/egress nodes are identified and included in NODES. A loop indexes links so that NTYPE for each link descriptor can be checked for a negative value. (A negative NTYPE value indicates that the node is an access/egress point.) If NTYPE is negative and the node has not been used before (as indicated by CHECK), it is added to NODES and the corresponding element in CHECK is flagged with a one.

Third, the tributary data is formulated for each NODES entry. A loop indexes the NODES entries and REFNOD is made equal to the node number contained in the indexed NODES entry. PATHS7 is called to identify all links adjacent to REFNOD. These links are stored in PATHS and are copied to elements 2-5 of the current NODES entry. If element 5 is zero (i.e. less than 4 links were identified by PATHS7), one link from elements 2-4 with a node elevation that is lower than the elevation of REFNOD is chosen and moved to element 5 (i.e. the link chosen must have a NODE2 elevation that is lower than the REFNOD elevation to become a 'to' link or a link to which water will travel because water in the sewer system is conveyed by gravitational forces). The NODE2 elevations of the remaining links in the elements 2-4 are also compared to the REFNOD elevation. If another valid link is found then OPTPTH is called so that the length of the path to an access/egress node for the current link and the link in element 5 can be determined. If the path associated with the current link is shorter than the path of the link

in element 5, the two links switch places; otherwise they stay where they are.

When the optimum 'to' link is in place, the other downsloping links are deleted from the 'from' positions (i.e. elements 2-4) These links actually represent different path choices, therefore the numbers of the link not chosen, the number of the link chosen, and the node at which the choice occurred are stored in CHOLNK. The remaining links in elements 2-4 are reorganized so that all 'from' links fill the first elements of NODES (i.e. a group with only one 'from' link will contain a zero in elements 3 and 4 and the 'from' link number in element 2). The loop then returns to calculate the group data for the next NODES entry.

Fourth, in the process of creating the data, some of the nodes originally included as possible centers of data may have become excluded from the system when various links were grouped with other nodes. The nodes excluded from the system are identified by looking for NODES entries having no 'from' links (i.e. elements 2, 3, and 4 in nodes equal zero) and deleted. When an entry is deleted, all successive NODES entries are shifted to fill the gap.

Fifth, the tributary data is reorganized so that the 'from' links precede all related 'to' links. CHECK is purged. The access/egress nodes are identified in NODES and stored in EXTREF. The index number of the NODES entry in which the access/egress node occurred is placed into the

array LIST1, and flags are set in CHECK to indicate which NODES groups have been used. For each tributary group represented by the LIST1 entries, the NODES information is copied to SRTIDX. A DO-WHILE loop is used to reorganize the tributary data and a loop indexes each LIST1 group. For each 'from' link in a LIST1 group, a related and unused 'to' link group is found in NODES and copied to SRTIDX; its index is stored in LIST2. Then another LIST1 group is indexed. When all the groups in LIST1 have been used, LIST2 is copied into LIST1 and the process is repeated.

When all the groups have been ordered, LIST2 will be empty, FLAG will be set to one, and the DO-WHILE loop will terminate.

Since SRTIDX is now organized so that the access/egress points are near the beginning of the list and all successive groups are near the end (i.e. opposite to what was needed), SRTIDX is copied in reverse order to TBTRY. However, only the 'from' and 'to' links are copied, since the node numbers are not required by succeeding routines. Control then passes to the calling program.

5.4.29 OPTPTH - A Routine to Determine the Length of the Optimum Path to an Access/Egress Point

The variables used by OPTPTH are:

* NTYPE - The type of node: cul-de-sac, dead-end, 3-way, etc. If a node is an access/egress node, then NTYPE will initially have a negative value; later, when the access/egress points no longer need to be identified, the negative values are converted to positive values.
* NODE1 - an array which holds the node of highest elevation for each link.
* NODE2 - an array which holds the node of lowest elevation for each link.
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* PCHIDX - index into POSCHN.
* POSCHN - an array containing the links that might belong to the chain currently being developed.
* CHNIDX - index into CHAINS.
* CHAINS - a two-dimensional array containing a list of chains and the links which make up each chain. A chain is a connected group of

links. Any chain that leads to an access/egress point is identified by a one in the zeroth element of the array.

* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being test, and then moves along all possible paths in an attempt to reach an access/egress point.
* BRANUM - the number of branches (forks in the path) encountered during a given continuity test of a link.
* BRALNK - an array holding the link number of the link which occurred before each branch.
* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the link numbers of all possible paths.
* FLAG1 - if set to one, signifies an end to the continuity test of a given link.
* FLAG2 - if set to one, signifies that REFLNK should return to the most recently encountered branch and try another path.
* CHECK - marks off the links that are obviously part of the system.

* RESULT - indicates whether the current path choice has been tried before. If RESULT=0, the current path choice has not been tried. If RESULT=1, the current path choice has been tried before and choosing this path would result in the duplication of an existing chain.
* LENGTH - the sum of the lengths of each individual link in a chain (m).
* OPTLEN - length of the optimum (shortest) path (m).
* NSL - number of system links.
* LINKS - links involved in the system.
* L,L1-3 - loop index variables.
* X,X1-2 - miscellaneous variables.

OPTPTH determines the optimum (shortest) path from a given link (as referenced by REFLNK) along downsloping links to an access/egress point.

The REFLNK pointer is moved in a manner that is similar to the method described in the continuity test section of IDDMSW. This routine considers downsloping links only when attempting to find a path to an access/egress point. Since easement links are a valid component of the domestic sewer system, PATHS4 is used to determine the possible path choices. Each time REFLNK points to a link, the link is stored in POSCHN; links are stored in the order they were

pointed to. When a dead end or an access/egress point is encountered or when REFLNK has pointed to 50 links, the contents of POSCHN are copied into CHAINS. (A one is stored in the zeroth element of CHAINS if an access/egress point was encountered.) The elements in POSCHN are then erased up to the last branch, a new chain is begun, and the procedure repeats. When all branches have been taken (i.e. all chains have been found), the shortest chain capable of reaching an access/egress point is identified.

A loop is used to step through the chains so that the chains ending in an access/egress point (i.e. the zeroth element equals one) may be found. Whenever such a chain is found, a second loop steps through the links in the chain and sums their lengths. The total length is stored in CHAINL in the element corresponding to the chain number. When the length of all chains ending in an access/egress point have been calculated, a loop is used to isolate the shortest chain. The procedure assumes that the first chain possesses the shortest length and stores the chain's number in X2. The length of each remaining chain is then compared to the length of chain X2. If a chain with a shorter length than chain X2 is discovered, the number of the shorter chain is stored in X2. When the loop ends, X2 identifies the shortest chain. The length of this chain is placed in OPTLEN and control returns to the calling program.

5.4.30 **PATHS - Overview**

Seven different but related routines with the generic description PATHSx, where X is a number from 1-7 are used. Each routine identifies nodes or links that are adjacent to a specified node or link.

PATHS1 is called by ERRCHK. PATHS1 identifies all unused links that are adjacent to a specified link.

PATHS2 is called by IDWATR, IDDMSW, GROUP1, and GROUP2. PATHS2 is very similar to PATHS1, because it also identifies all unused links that are adjacent to a specified link. However, PATHS2 is limited to choosing links from a given set of links belonging to the subsystem (i.e. water and domestic sewer systems) being evaluated.

PATHS3 is called by IDWATR. PATHS3 identifies all links adjacent to a specified link. PATHS3 ensures that chains are not duplicated. Easement links are not used. PATHS4 is called by IDDMSW and OPTPTH. PATHS4 is similar to PATHS3. However, PATHS4 considers the direction of flow and allows the use of easement links.

PATHS5 is called by TAGPIT and IDDMSW. PATHS5 identifies all nodes adjacent to a specified node.

PATHS6 is called by FNDLOP. PATHS6 identifies links adjacent to a specified link. PATHS6 ensures chains are not

duplicated. However, chains are allowed to form a closed loop. Easement links are not used.

PATHS7 is called by GROUP1 and GROUP2. PATHS7 identifies links adjacent to a specified node. The links are chosen from a given set of links.

5.4.30 (a) **PATHS1**

The variables used by PATHS1 are:

* NLINK - number of links in the subdivision.
* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.
* PATHS - an array listing the link numbers of all possible paths emanating from a specified link.
* L - loop index.
* X,X1-4 - miscellaneous variables.

PATHS1 identifies all unused links (as denoted by CHEKIN) that are adjacent to a specified link (REFLNK). A

loop compares the link descriptors (end nodes) of REFLNK with the link descriptors of all other links in the subdivision. When a link with the same end node as REFLNK is found, the corresponding element in CHECKIN is evaluated. If this element is zero, the link number is added to PATHS and NUMPTH is incremented. If the element is one, the link is ignored. Upon completion of the loop, PATHS and NUMPTH are returned to the calling routine.

5.4.30 (b) **PATHS2**

The variables used by PATHS2 are:

* NLINK - number of links in the subdivision.
* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.
* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the link numbers of all possible paths emanating from a specified link.
* LINKS - the links belonging to the subsystem being evaluated. (i.e., water and domestic sewer systems)
* NUMLNK - the number of links in the subsystem.
* L - loop index.

* X,X1-4 - miscellaneous variables.

PATHS2 identifies all unused links (as denoted by CHEKIN) belonging to the subsystem being evaluated (water and domestic sewer systems), that are adjacent to a specified link (REFLNK). A loop compares the link descriptors (end nodes) of REFLNK with the link descriptors of every link in the subsystem. When a link with the same end node as REFLNK is found, the corresponding element in CHEKIN is evaluated. If this element is zero, the link number is added to PATHS and NUMPTH is incremented. If the element is one, the link is ignored. Upon completion of the loop, PATHS and NUMPTH are returned to the calling routine.

5.4.30 (c) **PATHS3**

The variables used by PATHS3 are:

* NLINK - number of links in the subdivision.
* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.
* PCHIDX - index into POSCHN.
* POSCHN - an array containing the links possibly belonging to the chain currently being developed.
* CHNIDX - index into CHAINS.
* CHAINS - a two-dimensional array containing a list of chains and the links which make up each chain. (A chain is a connected group of links.) Any chain that leads to an

access/egress point is identified with a one in the zeroth element of the array.

* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the link number of all possible paths emanating from a specified link.
* RESULT - indicates whether the path choice to be made has been made before. If RESULT=0, the path choice has not been tried. If RESULT=1, the path has been tried.
* WROAD - an array containing the road width corresponding to each link. Values of zero indicate easement links (i.e. no road) (m).
* L - loop index.
* X,X1-4 - miscellaneous variables.

PATHS3 identifies all unused links (as denoted by CHEKIN) that are adjacent to a specified link (REFLNK). PATHS3 ensures that chains are not duplicated and easement links are not used. A loop indexes all links in the subdivision. When the link descriptors of the indexed link match the link descriptors of REFLNK, and PCHIDX is less than 10, the indexed link is tested for chain duplication. This test begins by adding the indexed link to POSCHN, and then comparing the elements of POSCHN to the corresponding elements of every chains via two nested loops. When POSCHN

and CHAINS correspond exactly, RESULT is set to one to indicate that the indexed link should not be used. RESULT is set to zero if POSCHN and CHAINS do not correspond exactly. In addition, the indexed link is added to PATHS and NUMPTH is incremented by one. Regardless of the value of RESULT, the indexed link is removed from POSCHN. Control then returns to the calling program.

5.4.30 (d) **PATHS4**

The variables used by PATHS4 are:

* NLINK - number of links in the subdivision.
* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* PCHIDX - index into POSCHN.
* POSCHN - an array containing the links possibly belonging to the chain currently being developed.
* CHNIDX - index into CHAINS.
* CHAINS - a two-dimensional array containing a list of chains and the links which make up each chain. (A chain is a connected group of links.) Any chain that leads to an access/egress point is identified with a one in the zeroth element of the array.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the

link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.

* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the link numbers of all possible paths emanating from a specified link.
* RESULT - indicates whether the path choice to be made has been made before. If RESULT=0, the path choice has not been tried. If RESULT=1, the path has been tried.
* L - loop index.
* X1-4 - miscellaneous variables

PATHS4 identifies links adjacent to a given link REFLNK. PATHS4 ensures that the links chosen do not duplicate existing chains and that they are downsloping relative to REFLNK. PATHS4 allows the use of easements.

A loop indexes all links in the subdivision. The node with the lowest elevation of REFLNK is compared to the node with the highest elevation of the indexed link. If the node numbers are the same and PCHIDX is less than or equal to 10, the indexed link is tested for chain duplication. This test relies on two nested loops which compare the elements of POSCHN to the corresponding elements of CHAINS. When POSCHN

and CHAINS correspond exactly, RESULT is set to one to indicate that the indexed link should not be used. RESULT is set to zero if POSCHN and CHAINS do not correspond exactly. In addition, the indexed link is added to PATHS and NUMPTH is incremented by one. Regardless of the value of RESULT, the indexed link is removed from POSCHN. Control then returns to the calling program.

5.4.30 (e) **PATHS5**

The variables used by PATHS5 are:

* NLINK - number of links in the subdivision.
* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* REFNOD - node currently being referenced.
* PATHS - an array listing the node numbers of all possible path choices emanating from the specified node.
* L - loop index.
* X1-4, - miscellaneous variables.

PATHS5 identifies all nodes adjacent to a given node, REFNOD. A loop indexes links in the subdivision so that the link descriptors of each link can be compared to REFNOD. When a match is detected, the node belonging to the indexed link and adjacent to REFNOD is added to PATHS and NUMPTH is incremented by one. Control then returns to the calling routine.

5.4.30 (f) **PATHS6**

The variables used by PATHS6 are:

* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* CHEKIN - an array which keeps track of which links have been tested for a direct or indirect connection to an access/egress point.
* REFLNK - a pointer to the link being evaluated by the test routine (testing for continuity with an access/egress point) at any given time. REFLNK (reference link) starts at the link being tested, and then moves along all possible paths in an attempt to reach an access/egress point.
* NUMPTH - number of possible paths which REFLNK can follow from a given link.
* PATHS - an array listing the node numbers of all possible paths emanating from a specified link.
* LENGTH - the sum of the lengths of each individual link in a chain (m).
* PCHIDX - index into POSCHN.
* POSCHN - possible chain currently being checked.

* CHNIDX - index into CHAINS.
* CHAINS - a two-dimensional array containing a list of chains and the links which make up each chain. (A chain is a connected group of links.) Any chain that leads to an access/egress point is identified by a one in the zeroth element of the array.
* RESULT - indicates whether the current path choice has been tried before.
* ENNODE - the last node in the chain; the node to which new links should be attached.
* CSLEN - current shortest length (m).
* LLINK - link length (m).
* WROAD - an array containing the road width corresponding to each link in the subdivision (m).
* WLINKS - links in the watermain system.
* L - loop index.
* X1,X2, - miscellaneous variables.

PATHS6 identifies links adjacent to a given link, REFLNK The links are chosen from the set of watermain links. PATHS6 ensures that chains are not duplicated and that previously used links are not used again unless the link forms a complete loop in POSCHN. (Loops are desirable for future Hardy Cross analysis.)

First, a loop is used to sum the lengths of the links in POSCHN. The total length is stored in LENGTH. Another loop indexes each link in the watermain system. Then the last node of the last link in POSCHN is identified. If POSCHN is just beginning then either end of the link in POSCHN can be considered to be the last node in the chain. The last node is then compared to the link descriptors of the indexed watermain link.

If the indexed link is connected to the last node and if PCHIDX is less than 50 (the maximum length of the loop), and LENGTH is less than CSLEN, then the link is also tested for chain duplication. This test relies on two nested loops which compare the elements of POSCHN to the corresponding elements of CHAINS. When POSCHN and CHAINS correspond exactly, RESULT is set to one, otherwise RESULT equals zero. Additional tests on the indexed link ensure that RESULT=0, that the link hasn't been used before (CHEKIN=0), and that the link is not an easement (WROAD>0). If the indexed link passes all tests, it is added to PATHS and NUMPTH is incremented by one.

If PCHIDX is greater than 2 (it takes three links to make a loop), the indexed link is tested for loop formation. (Since the first link would be marked as 'used' in CHEKIN, the remainder of this routine disregards the 'not used before' criteria.) A loop is formed when the first link in POSCHN is used twice and when the last link is attached to

the first link correctly (i.e. If the first link has nodes A and B, and B is the common node between the first and second links, then A must be the node that is common between the first and last links. If this were not the case, a loop could be formed where the first link is only attached to the loop by one node, rather than being a segment of the loop.) If a valid loop is formed, the indexed link is added to PATHS and NUMPTH is incremented by one.

When all links in the watermain system have been tested, control returns to the calling program.

5.4.30 (g) **PATHS7**

The variables used by PATHS7 are:

* NODE1 - an array which holds the node of highest elevation for each link (m).
* NODE2 - an array which holds the node of lowest elevation for each link (m).
* PATHS - an array listing the link numbers of all possible paths.
* REFNOD - node currently being referenced.
* NSL - number of system links.
* LINKS - the links belonging to the subsystem being evaluated.
* NUMPTH - number of possible path choices emanating from the links associated with REFNOD.
* PATHS - an array listing the link numbers of all possible path choices emanating from the specified node.
* L - loop index.
* X1,X2, - miscellaneous variables.

PATHS7 identifies links adjacent to a given node, REFNOD. The links are chosen only from the set of links defined by LINKS. A loop indexes links so that the link descriptors of each link can be compared to REFNOD. When a match is found, the indexed link is stored in PATHS and NUMPTh is incremented by one. Upon completion of the loop, control returns to the calling program.

5.4.31 SAC - A Routine to Calculate the Pavement Area of a Cul-de sac

The variables used by SAC are:

* ASAC - area of the road (m^2).
* LEN - length of the road (m).
* NODE1 - origin node for the current link.
* CNODE - current node.
* ACENOD - the access link node to the cul-de-sac.
* NODE2 - destination node for the current link.
* ATTLIN - links associated with the specified node.
* ACELIN - the access link to the cul-de-sac.
* WROAD - width of road of a given link (m).
* IRAD - island radius (m).
* RADIUS - radius of the cul-de-sac or tube sac (m).
* CULSAC - half of the cul-de-sac pavement area (m^2).
* TRP - the other half of the cul-de-sac pavement area not taking into account the island (m^2).
* LLINK - link length (m).
* PRAD - peak radius for obelisk island (m).
* K - the height of the trapezoid that is assumed to represent half of an obelisk island (m).
* P1 - is half of the island area of a circle island (m^2).
* P2 - area of the trapezoid (m^2).

* TV - the radius of the cul-de-sac used to measure its pavement length (m).
* WITH1 - width of the link accessing the cul-de-sac (m).

The SAC routine calculates the cul-de-sac pavement area. The routine considers the access link and island within the cul-de-sac.

First the routine identifies the two nodes attached to the link leading to the cul-de-sac. This is done by checking to see if CNODE, which is the node of NTYPE equal to 1, is NODE1 of I, the current link passed from PAVEMENT. If so ACENOD is NODE2(I), if not ACENOD is NODE1(I).

Next SAC checks the ATTLIN of ACENOD. The ATTLIN that is not either equal to 0 or I becomes ACELIN. WITH1, becomes the WROAD of ACELIN, the access link to the cul-de-sac.

The routine then starts the area calculations (see figure 8). SAC calculates the island radius:

IRAD = RADIUS (CNODE) - WROAD (I)

The dead-end area of the cul-de-sac

$$CULSAC = 0.5 * [ACOS(-1.0)] * \{[RADIUS(CNODE)]^2 - IRAD^2\}$$

The entrance to the cul-de-sac.

TRP = 0.5 * LLINK(I) * [2.0 * RADIUS(CNODE) + WITH1]

The entrance of the cul-de-sac area is adjusted by P1,P2 to consider an obelisk size island. If LLINK equals 0 PRAD equals IRAD and the island is a circle. ANG is calculated as:

ANG = ARCTAN [IRAD / LLINK (I)] * 2

If ANG is greater than 90 degrees, PRAD = IRAD. If ANG is less than 90 PRAD is the value of 0.5 and K is calculated:

K = LLINK(I) - LLINK (I) * PRAD / IRAD

$P1 = 0.5 * PI * PRAD^2$

P2 = 0.5 * K * (2.0 * IRAD + 2.0 * PRAD)

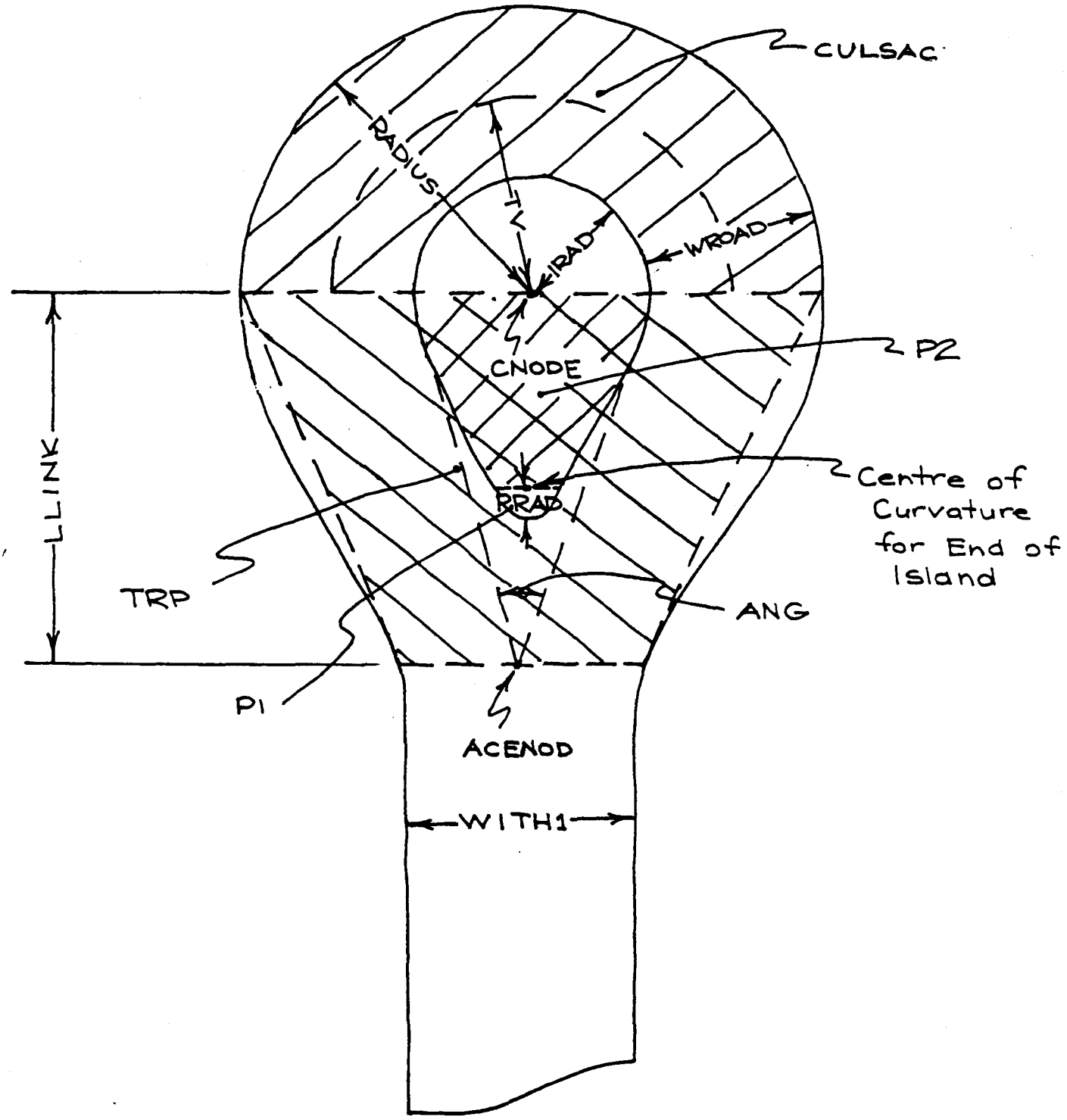
ASAC = CULSAC + TRP + P1 + P2

TV = IRAD + WROAD(I) / 2.0

$LEN = TV * PI + 2.0 * [LLINK(I)^2 + TV^2]^{1/2}$

Control is then returned to PAVEMENT.

FIGURE 8 - VARIABLES USED IN SAC ROUTINE



5.4.32 SSSDPTH - A Routine to Allocate Storm Sewer Elevations to Nodes

The variables used by SSSDPTH are:

* ATTLIN - links associated with the specified node.
* CLINK - current link.
* CNODE - current node in memory.
* SSLINK - storm sewer elevation of a given link (m).
* NODE1 - origin node of the current link.
* NODE2 - destination node for the current link.
* DPTH1 - the depth of excavation at the origin node (m).
* DPTH2 - the depth of excavation at the destination node (m).
* ELEV - ground elevation of the node (m).

The SSSDPTH routine is called by the STORM routine to assign the correct elevation and excavation depths to the nodes for storm sewer pipe. CLINK, DPTH1, DPTH2 are the variables being passed to SSSDPTH.

Each link associated with a certain node is checked to see if it is the CLINK being passed. If so then the elevation is calculated as being the ELEV-DPTH1. An identical calculation is performed for the destination and origin nodes.

5.4.33 SWTABLE - A Routine to Create the Sewer/Watermain Output Table

The variables used by SWTABLE are:

* YSEC - number of rows in each section of the link description tables.
* XSEC - number of sections in the link description tables.
* XSPA - number of columns in each section of the link description tables.
* MCOL - maximum number of columns in the link description tables.
* COL - starting column for the link description tables.
* LENGTH - length of the table section subheadings.
* IFRONT - starting column for the table section subheadings.
* PIPLEN - total subdivision sewer pipe length (m).
* YTOTAL - flag to indicate when a table line division should be printed.
* NLINK - number of links.
* PIPE - sewer pipe length per link (m).
* LLINK - link length (m).
* UCOST - sewer unit pipe cost.
* TCOST - sewer pipe cost per link.
* DIA - sewer pipe diameter per link (mm).

The SWTABLE routine writes the domestic and storm sewer, watermain; pipe diameters, lengths and costs. This routine is called by the ANALYSIS routine. The routine first outputs the table format. Then each link is processed to determine PIPE, and TCOST. This is done by checking the DIA of each link. If DIA is greater than 0 then PIPE equals LLINK of the link and PIPELIN is incremented by LLINK. These results are output in the table.

5.4.34 TAGPIT - A Routine to Determine Whether a Node is a Pit With Inflows but no Outflows

The variables used by TAGPIT are:

* NTYPE - the type of node: cul-de-sac, dead-end, 3-way, etc. If a node is an access/egress node, then NTYPE will initially have a negative value; later, when the access/egress points no longer need to be identified, the negative values are converted to positive values.
* NODE1 - an array which holds the node of highest elevation for each link.
* NODE2 - an array which holds the node of lowest elevation for each link.
* ELEV - an array which holds the natural ground elevation of each node (m).
* NUMPTH - number of possible paths leading from a specified link.
* PATHS - an array listing the link numbers of all possible paths.
* PITFLG - indicates whether the specified node is a pit. PITFLG=0 if the node is not a pit. PITFLG=1 if the node is a pit.

TAGPIT determines whether the specified node, REFNOD, is a pit (i.e. a node that is lower than all surrounding nodes with inflows but no outflows). First, TAGPIT locates all nodes immediately adjacent to the specified node by calling PATHS5. These nodes are stored in the array PATHS. Then a loop is used to index these nodes so that their ground elevations may be compared with the reference node. When a node with a higher elevation than REFNOD is encountered, the number of paths (NUMPTH) is decremented by one. Upon completion of the loop, if NUMPTH equals zero, PITFLG is set to one indicating that the ground elevation of REFNOD is lower than all other immediately adjacent nodes. If NUMPTH is greater than zero, PITFLG is set to zero. Control then returns to the calling routine.

5.4.35 THREEWAY - A Routine to Identify the Threeway Intersection Locations and to Calculate their Extra Lengths for LIGHT

The variables used by THREEWAY are:

* EXRLEN - extra street length (m).
* NTYPE - node type.
* ORIGIN - the origin node of the sequence of nodes comprising a street. (One street may be composed of several links.)
* DEST - the destination node of a sequence of nodes comprising a street. (One street may be composed of several links.)
* LOPCON - loop controller.
* CNODE - current node.
* RATIO - the street length ratio.
* LAST - the last link between two intersections or an intersection and a dead-end, or an intersection and a cul-de-sac in the current branch.
* BRANCH - one of a number of links attached to the node.
* CLINK - current link.
* TROAD - type of road or street.

* PATH - the links between two intersections or intersection and a dead-end, or an intersection and a cul-de-sac.
* ATTLIN - links associated with the specified node.
* LAST - the last link between two intersections or intersection and a dead-end, or an intersection and a cul-de-sac.
* DIRECT - indicates the origin or destination node of a current branch.
* EXIT - exit node.
* CORWD - corresponding road width (m).
* LINK - destination or origin link.
* NODE1 - origin node for the current link.
* NODE2 - destination node for the current link.

The THREEWAY routine checks for the existence of threeway intersections at the various nodes. First THREEWAY identifies the node as CNODE and the number of threeway intersections at the node. The ORIGIN and DEST NTYPE's are checked to see if they are both of type 3. If so then LOPCON equals 2 and CNODE equals ORIGIN. If either ORIGIN and DEST NTYPE's are of type 3, LOPCON equals 1. If NTYPE(ORIGIN) equals 3 then CNODE equals the ORIGIN, NTYPE(DEST) equals 3 then CNODE equals DEST. If neither DEST or ORIGIN have a NTYPE of 3 LOPCON equals 0.

If LOPCON is 2 indicating that the street has two threeway intersections, CNODE equals DEST and RATIO equals 0.5. If LOPCON is 1 then RATIO is 1.0. If LAST equals 1 and the NTYPE's of the ORIGIN and DEST nodes equal 3 or 4 then RATIO equals 0.5. The street length ratios are calculated and used in adjusting the street lengths by the appropriate fraction of the intersection. This is done for the appropriate light placements.

THREEWAY identifies the origin and destination links for the calculation of the extra street lengths for street light allocation. If ORIGIN is not the same as DEST, CLINK is assigned the first link and then the last link that is a part of the PATH between ORIGIN and BRANCH. Then LINK is assigned to the ATTLIN of the CNODE equal to CLINK. If ORIGIN equals DEST and LOPCON equals 1 LINK is the first link that is a part of the PATH between ORIGIN and BRANCH. If LOPCON equals 0 LINK is the last link that is a part of the PATH between ORIGIN and BRANCH.

Once LINK is assigned, DIRECT equals 1 if NODE1(LINK) equals CNODE. In other words the origin nodes of the LINK and the BRANCH passed from LIGHT are the same. If they are not the same then DIRECT equals 2. If EXIT(CNODE) equals 0 then EXRLEN from the origin and/or destination nodes is:

EXRLEN = EXRLEN + RATIO * CORWD (LINK,DIRECT)

The routine outputs ORIGIN, DEST, CNODE, LINK information and returns control to LIGHT.

5.4.36 TUBE - A Routine to Calculate the Pavement Area of the Tube-Sac

The variables used by TUBE are:

* PI - arc cosine of -1.0 (rad).
* ASAC - area of the road (m^2).
* LEN - length of the pavement in the tube-sac (m).
* IRAD - island radius (m).
* TV - link length in the tube taking into account the island width adjustments (m).
* LLINK - link length (m).
* ADJWD - island width adjustments to the tube link length equivalent to CORWD (m).
* D - island length (m).
* RADIUS - radius at the end of the tube sac (m).
* CNODE - current node.
* ISLAND - area of the island (m^2).
* WROAD - width of road of a given link (m).

The TUBE routine calculates the pavement area required for the tube-sacs in the subdivision. The routine first calculates the road length around the island and the length of the island itself. This is done by first specifying the minimum island radius as being equal to 0.5 meters. Then TV is calculated as being the LLINK minus the width of the road associated with the tube-sac entrance node. (See figure 9).

Next TUBE calculates the D, the island length. It is assumed that TV minus D is twice the RADIUS minus the IRAD. The island size is therefore the area of a rectangle and two semi-circles.

$$ISLAND = (IRAD * PI)^2 + 2.0 * IRAD * D$$

If the WROAD passed to TUBE is less than RADIUS:

$$ASAC = 0.5 * RADIUS(CNODE) * PI^2 + TV * WROAD - ISLAND.$$

Otherwise:

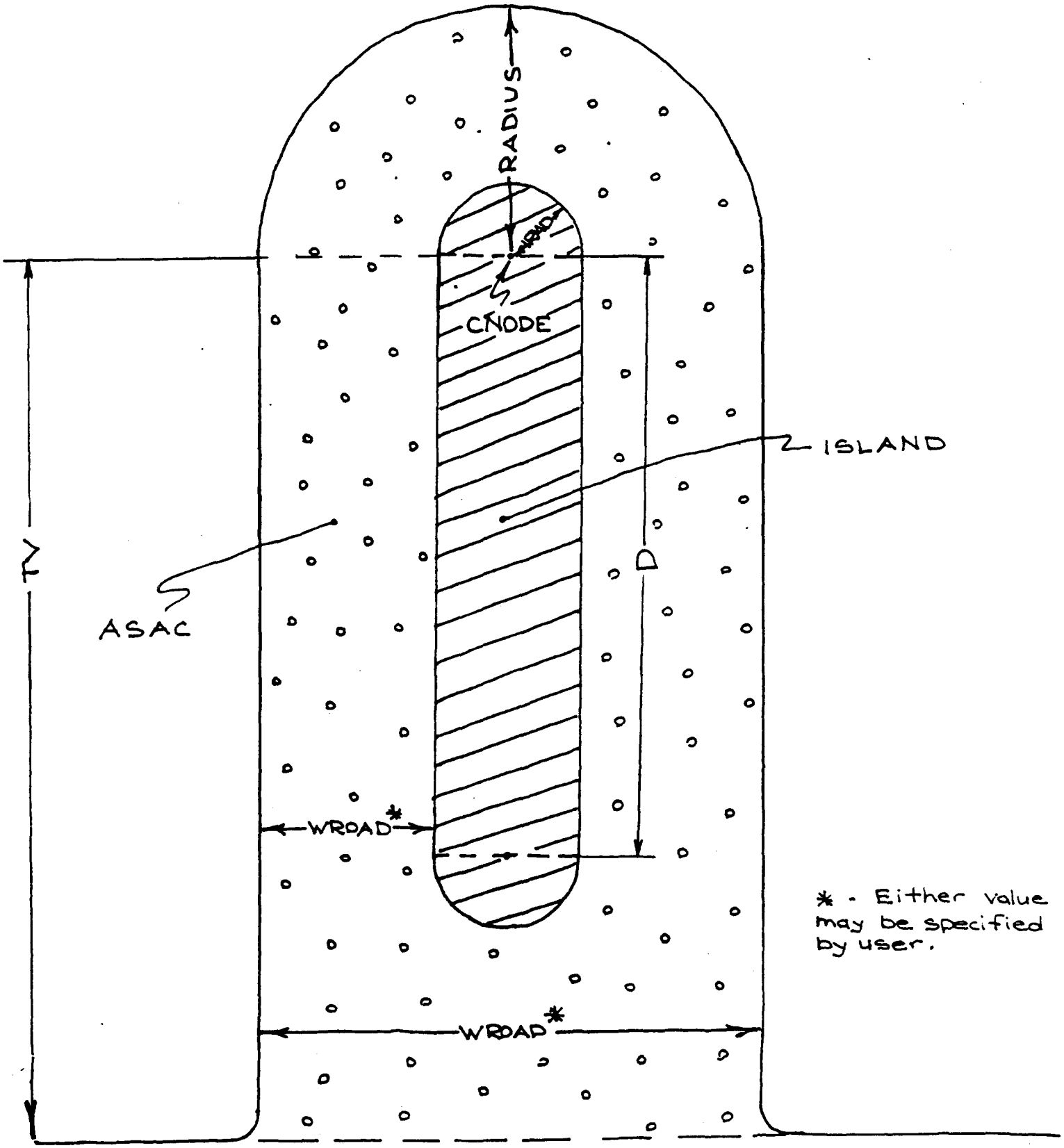
$$ASAC = 0.5*RADIUS(CNODE)*PI^2+TV*2*(WROAD+IRAD)-ISLAND.$$

Finally LEN is calculated

$$LEN = TV * 2.0 + RADIUS(CNODE)*PI$$

The results are returned to PAVEMENT.

FIGURE 9 - VARIABLES USED IN TUBE ROUTINE



* - Either value may be specified by user.

5.5 DESCRIPTION OF RACAM FUNCTIONS

5.5.1 ARTIHA,ARITHP,ARITHF - Functions to Calculate the Arithmetic Gradient of Annual ,Present and Future Costs.

Variables used by ARITHA are:

* ARITHA - annual cost calculation using present Value concepts.
* BASE - the capital amount to be adjusted by the present value factor and the rate.
* GRAD - growth factor, or gradient for annual costs.
* INTRST - interest rate used for present value calculations.
* RATE - decimal rate of INTRST.
* W - single payment compound amount factor. (SPCAF)
* N - number of years.

ARITHA, ARITHP and ARITHF are used to calculate annual,present and future costs of various subdivision elements, respectively. All three functions base their calculations on the growing perpetuity present value financial model.

$$W = (1.0 + RATE)^N$$

$$ARITHA = BASE + GRAD * [1.0 / RATE - N / (W - 1.0)]$$

$$ARITHF = GRAD/RATE*[(W-1.0)/RATE-N]+BASE*(W-1.0)/RATE$$

```
ARITHP = GRAD/RATE*[(W-1.0)/(RATE*W)-N/W]+BASE*(W-
1.0)/(RATE*W)
```

5.5.2 UNFORM,UNFORM1,UNFORM2 - Functions that Calculate the Present, Annual and Future Values of an Expenditure.

The variables used by UNFORM(1,2) are:

* RATE - interest rate in decimal.
* INTRST - interest rate in percentage.
* W - present value factor.
* N - life of the project.
* PWORTH - the capital amount to be adjusted by the present value factor and the rate.
* AWORTH - the annual expenditure to be discounted back to present values.

The UNFORM functions are called by the financial routines to calculate present and future values of expenditures and to calculate the annuity values for various expenditures and recurring costs.

$$W = (RATE + 1.0)^N$$

$$UNFORM = PWORTH * [RATE * W / (W - 1.0)]$$

$$UNFORM1 = AWORTH * [(W - 1.0) / (RATE * W)]$$

$$UNFORM2 = PWORTH * (1.0 + RATE)^N$$

APPENDIX A

DATA ACQUISITION SHEETS

This appendix contains data sheets used to compile data required for RACAM input data files. There are two sets of data sheets: those for the economic data, and those for the link and node data.

APPENDIX A SAMPLE DATA ACQUISITION SHEETS

A.1 ECONOMIC DATA SHEETS

A.1.1 SAMPLE DATA SHEET FOR INTEREST RATE AND SUBDIVISION LIFE

INTEREST RATE AND SUBDIVISION LIFE

SUBDIVISION:

SHEET #:

DATE:

INTEREST RATE	SUBDIVISION LIFE

A.1.2 SAMPLE DATA SHEET FOR STREET REQUIREMENT UNIT COSTS

STREET REQUIREMENT UNIT COSTS

SUBDIVISION:

SHEET #:

DATE:

STREET REQUIREMENT	UNIT COST
STREET SIGN	($/SIGN)
STREET LIGHT	($/LIGHT)
PARK DEVELOPMENT	($/HECTARE)
BUFFER DEVELOPMENT	($/HECTARE)
COLLECTOR STREET	($/$m^2$)
LOCAL STREET	($/$m^2$)
SIDEWALK	($/$m^2$)
BARRIER CURB & GUTTER	($/m)
ROLLED CURB & GUTTER	($/m)
BOULEVARD CURB & GUTTER	($/m)

A.1.3 SAMPLE DATA SHEET FOR WATERMAIN UNIT COSTS

WATERMAIN UNIT COSTS

SUBDIVISION:

SHEET #:

DATE:

DIAMETER INDEX	DIAMETER RANGE	UNIT COST ($/m)
1	0 - 150 mm	
2	151 - 200 mm	
3	201 - 250 mm	
4	251 - 300 mm	
5	301 - 400 mm	
6	401 - 9999 mm	

A.1.4 SAMPLE DATA SHEET FOR DOMESTIC SEWER UNIT COSTS

DOMESTIC SEWER UNIT COSTS

SUBDIVISION:

SHEET #:

DATE:

* DIAMETER INDEX	** DEPTH INDEX	UNIT COST ($/m)
1	1	
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	
1	8	
1	9	
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	
2	8	
2	9	

3	1	
3	2	
3	3	
3	4	
3	5	
3	6	
3	7	
3	8	
3	9	
4	1	
4	2	
4	3	
4	4	
4	5	
4	6	
4	7	
4	8	
4	9	
5	1	
5	2	
5	3	
5	4	
5	5	
5	6	
5	7	

5	8	
5	9	

* DIAMETER INDEX	DIAMETER RANGE
1	0 - 200 mm
2	201 - 250 mm
3	251 - 300 mm
4	301 - 375 mm
5	greater than 375 mm

** DEPTH INDEX	DEPTH RANGE
1	0 - 3.0 m
2	3.1 - 3.5 m
3	3.6 - 4.0 m
4	4.1 - 4.5 m
5	4.6 - 5.0 m
6	5.1 - 5.5 m
7	5.6 - 6.0 m
8	6.1 - 6.5 m
9	Out of Bounds

A.1.5 SAMPLE DATA SHEET FOR STORM SEWER UNIT COSTS

STORM SEWER UNIT COSTS

SUBDIVISION:

SHEET #:

DATE:

* DIAMETER INDEX	** DEPTH INDEX	UNIT COST ($/m)
1	1	
1	2	
1	3	
1	4	
1	5	
1	6	
1	7	
1	8	
1	9	
2	1	
2	2	
2	3	
2	4	
2	5	
2	6	
2	7	
2	8	
2	9	
3	1	

3	2	
3	3	
3	4	
3	5	
3	6	
3	7	
3	8	
3	9	
4	1	
4	2	
4	3	
4	4	
4	5	
4	6	
4	7	
4	8	
4	9	
5	1	
5	2	
5	3	
5	4	
5	5	
5	6	
5	7	
5	8	

5	9	
6	1	
6	2	
6	3	
6	4	
6	5	
6	6	
6	7	
6	8	
6	9	
7	1	
7	2	
7	3	
7	4	
7	5	
7	6	
7	7	
7	8	
7	9	
8	1	
8	2	
8	3	
8	4	
8	5	
8	6	

8	7	
8	8	
8	9	
9	1	
9	2	
9	3	
9	4	
9	5	
9	6	
9	7	
9	8	
9	9	
10	1	
10	2	
10	3	
10	4	
10	5	
10	6	
10	7	
10	8	
10	9	
11	1	
11	2	
11	3	
11	4	

11	5	
11	6	
11	7	
11	8	
11	9	
12	1	
12	2	
12	3	
12	4	
12	5	
12	6	
12	7	
12	8	
12	9	
13	1	
13	2	
13	3	
13	4	
13	5	
13	6	
13	7	
13	8	
13	9	
14	1	
14	2	

14	3	
14	4	
14	5	
14	6	
14	7	
14	8	
14	9	

* DIAMETER INDEX	DIAMETER RANGE
1	0 - 300 mm
2	301 - 375 mm
3	376 - 450 mm
4	451 - 525 mm
5	526 - 600 mm
6	601 - 675 mm
7	676 - 750 mm
8	751 - 900 mm
9	901 - 1050 mm
10	1051 - 1200 mm
11	1201 - 1350 mm
12	1351 - 1500 mm
13	1501 - 1650 mm
14	1651 - 1800 mm

** DEPTH INDEX	DEPTH RANGE
1	0 - 3.0 m
2	3.1 - 3.5 m
3	3.6 - 4.0 m
4	4.1 - 4.5 m
5	4.6 - 5.0 m
6	5.1 - 5.5 m
7	5.6 - 6.0 m
8	6.1 - 6.5 m
9	Out of Bounds

A.1.6 SAMPLE DATA SHEET FOR MANHOLE AND CATCHBASIN UNIT COSTS

MANHOLE AND CATCHBASIN UNIT COSTS

SUBDIVISION:

SHEET #:

DATE:

MANHOLE OR CATCHBASIN	UNIT COST
MANHOLE BASE,COVER,RISER,FRAME	($/MANHOLE)
MANHOLE SHAFT	($/m-DEPTH)
CATCHBASIN FRAME,COVER	($/CATCHBASIN)
CATCHBASIN SHAFT	($/m-DEPTH)

A.1.7 SAMPLE DATA SHEET FOR CATCHBASIN LEAD UNIT COSTS

CATCHBASIN LEAD UNIT COSTS

SUBDIVISION:

SHEET #:

DATE:

DEPTH INDEX	DEPTH RANGE	UNIT COST ($/m)
1	0.0 - 3.0 cm	
2	3.1 - 3.5 cm	
3	3.6 - 4.0 cm	
4	4.1 - 4.5 cm	
5	4.6 - 5.0 cm	
6	5.1 - 5.5 cm	
7	5.6 - 6.0 cm	
8	6.1 - 6.5 cm	
9	Out of Bounds	

A.1.8 SAMPLE DATA SHEET FOR DWELLING AND MAINTENANCE UNIT COSTS

DWELLING AND MAINTENANCE UNIT COSTS

SUBDIVISION:

SHEET #:

DATE:

DWELLING OR MAINTENANCE	UNIT COST
POWER-SINGLE FAMILY UNIT	($/UNIT)
POWER-MULTI FAMILY UNIT	($/UNIT)
GAS-SINGLE FAMILY UNIT	($/UNIT)
GAS-MULTI FAMILY UNIT	($/UNIT)
PHONE-SINGLE FAMILY UNIT	($/UNIT)
PHONE-MULTI FAMILY UNIT	($/UNIT)
STREET SWEEPING	($/Km/YR)
WINTER ROAD MAINTENANCE	($/Km/YR)
ASPHALT MAINTENANCE	($/Km/YR)
CONCRETE MAINTENANCE	($/Km/YR)
SEWER MAINTENANCE	($/Km/YR)
WATER DISTRIBUTION	($/Km/YR)
SOLID WASTE COLLECTION	($/HECTARE/YR)
STREET LIGHT MAINTENANCE	($/LIGHT/YR)
PARK & BUFFER MAINTENANCE	($/HECTARE/YR)

A.1.9 SAMPLE DATA SHEET FOR RACAM CALIBRATION VALUES

RACAM CALIBRATION VALUES

SUBDIVISION:

SHEET #:

DATE:

CONSTANT	* VALUE
POPULATION DENSITY	(PEOPLE/UNIT)
WATERMAIN MINIMUM VELOCITY	(m/s)
MANNING'S ROUGHNESS	
DOMESTIC WATER USE	(m^3/PERSON/DAY).
WATER USE SAFETY FACTOR	
DOMESTIC SEWER COVER DEPTH	(m)
DOMESTIC SEWER MIN. VELOCITY	(m/s)
DOMESTIC SEWER MAX. VELOCITY	(m/s)
STORM SEWER COVER DEPTH	(m)
STORM SEWER MIN. VELOCITY	(m/s)
STORM SEWER MAX. VELOCITY	(m/s)
MANHOLE SPACING	(m)
STREET LIGHT SPACING	(m)

Note: Entering "0.0" for any of these values will cause default values to be assumed.

A.1.10 SAMPLE DATA SHEET FOR ACCOUNT NAMES AND NUMBERS

ACCOUNT NAMES AND NUMBERS

SUBDIVISION:

SHEET #:

DATE:

*ACCOUNT #	**ACCOUNT NAME

*** MAY BE UP TO 12 CHARACTERS IN LENGTH**
**** MAY BE UP TO 35 CHARACTERS IN LENGTH**

A.1.11 SAMPLE DATA SHEET FOR ACCOUNT PERCENTAGES BY SUBDIVISION

ACCOUNT PERCENTAGES BY SUBDIVISION ELEMENT

SUBDIVISION:

SHEET #:

DATE:

*ELEMENT	# OF ACCOUNTS	ACCOUNT #	PERCENTAGE

* ENTER SUBDIVISION ELEMENTS AND THEIR RESPECTIVE ACCOUNT DATA IN THE FOLLOWING ORDER:

1. WATERMAIN
2. DOMESTIC SEWER
3. STORM SEWER
4. PAVEMENT
5. SIDEWALK
6. BARRIER CURB AND GUTTER
7. ROLLED CURB AND GUTTER
8. BOULEVARD CURB AND GUTTER
9. PARK DEVELOPMENT
10. BUFFER DEVELOPMENT
11. CATCHBASIN
12. MANHOLE
13. STREET LIGHT
14. STREET SIGN
15. TRAFFIC SIGN
16. GAS - SINGLE-FAMILY
17. POWER - SINGLE-FAMILY
18. PHONE - SINGLE-FAMILY
19. GAS - MULTI-FAMILY
20. POWER - MULTI-FAMILY
21. PHONE-MULTI-FAMILY

A.1.12 SAMPLE DATA SHEET FOR MAINTENANCE COSTS BY SUBDIVISION ELEMENT

MAINTENANCE COSTS BY SUBDIVISION ELEMENT

SUBDIVISION:

SHEET #:

DATE:

ELEMENT #	COST	GRADIENT	INTEREST	LIFE
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				

19				
20				
21				

A.1.13 SAMPLE DATA SHEET FOR OPERATION COSTS BY SUBDIVISION ELEMENT

OPERATION COSTS BY SUBDIVISION ELEMENT

SUBDIVISION:

SHEET #:

DATE:

ELEMENT #	COST	GRADIENT	INTEREST	LIFE
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				

18				
19				
20				
21				

A.2 LINK AND NODE DATA SHEETS

A.2.1 SAMPLE DATA SHEET FOR NODE DATA

NODE DATA

SUBDIVISION:

SHEET #:

DATE:

* NODE TYPE	NODE ELEVATION	RADIUS

* NODE TYPE INTERSECTION

1	TERMINAL
2	DIRECTIONAL
3	3-WAY STREET INTERSECTION
4	4-WAY STREET INTERSECTION
5	OTHER

ENTER A MINUS SIGN ("-")IN FRONT OF THE NODE TYPE FOR SUBDIVISION ACCESS NODES

A.2.2 SAMPLE DATA SHEET FOR LINK DATA I

LINK DATA I

SUBDIVISION:

SHEET #:

DATE:

ORIGIN	DEST.	LENGTH	SINGLE	MULTI	AREA	FRNTGE

NOTE:

ORIGIN = LINK ORIGIN NODE
DEST. = LINK DESTINATION NODE
LENGTH = LINK LENGTH
SINGLE = # OF SINGLE-FAMILY DWELLINGS ON LINK
MULTI = # OF MULTI-FAMILY DWELLINGS ON LINK
AREA = DRAINAGE AREA FOR LINK
FRNTGE = LINK FRONTAGE

A.2.3 SAMPLE DATA SHEET FOR LINK DATA II

LINK DATA II

SUBDIVISION:

SHEET #:

DATE:

ROAD LENGTH	ROAD WIDTH	ROAD TYPE

A.2.4 SAMPLE DATA SHEET FOR LINK DATA III

LINK DATA III

SUBDIVISION:

SHEET #:

DATE:

SIDEWALK WDTH	BARRIER LNGTH	ROLLED LNGTH	BLVD LNGTH

NOTE:

SIDEWALK WDTH = WIDTH OF SIDEWALK FOR LINK
BARRIER LNGTH = LENGTH OF BARRIER CURB AND GUTTER FOR LINK
ROLLED LNGTH = LENGTH OF ROLLED CURB AND GUTTER FOR LINK
BLVD LNGTH = LENGTH OF BOULEVARD CURB AND GUTTER FOR LINK

A.2.5 SAMPLE DATA SHEET FOR LINK DATA IV

LINK DATA IV

SUBDIVISION:

SHEET #:

DATE:

LINK #	W POPULATION	DS POPULATION	SS AREA(Ha)

NOTE:

W POPULATION = INITIAL POPULATION OF LINK FOR WATERMAIN CALCULATIONS

DS POPULATION = INITIAL POPULATION OF LINK FOR DOMESTIC SEWER CALCULATIONS

SS AREA = INITIAL DRAINAGE AREA (IN HA.) OF LINK FOR STORM SEWER CALCULATIONS

A.2.6 SAMPLE DATA SHEET FOR SUBDIVISION AREAS

SUBDIVISION AREAS

SUBDIVISION:

SHEET #:

DATE:

AREA DESIGNATION	AREA (HA.)
PARK	
BUFFER	
SCHOOL	
MULTI-FAMILY DWELLINGS	
COMMERCIAL	
OTHER	
TOTAL SUBDIVISION AREA	

APPENDIX B

PROGRAM LISTING

This appendix contains the FORTRAN code for each module of the RACAM program. The listing for the RACAM main routine appears first, followed by the subroutines called by the main routine. The subroutines called by other subroutines then appear, followed by functions used by RACAM. Each of the modules in these four sections are listed in alphabetical order.

APPENDIX B PROGRAM LISTING

B.1 MAIN ROUTINE: RCMAIN

```
*****************************************************************
*                                                               *
*            RESIDENTIAL AREA COST ANALYSIS MODEL (RACAM)       *
*                                                               *
*                    [ Regina, Saskatchewan ]                   *
*                                                               *
*     THIS PROGRAM WILL ITEMIZE AND CALCULATE THE VARIOUS COSTS *
*     ASSOCIATED WITH THE CONSTRUCTION OF A RESIDENTIAL SUBDIVISION *
*     AND PERFORM AN ECONOMIC ANALYSIS.                         *
*                                                               *
*                    UNIVERSITY OF REGINA                       *
*                                                               *
*                         AUGUST 1987                           *
*                                                          *
*                                                               *
*****************************************************************
* NINT   - number of interest rates.
* INTRST - the interest rates themselves.
* LIFE   - different lives of subdivision for cost analysis.
* NNODE  - number of nodes in the subdivision.
* NTYPE  - type of node.
* ELEV   - elevation of node.
* NLINK  - number of links in subdivision.
* NODE1  - high end node of link.
* NODE2  - low end node of link.
* LLINK  - length of link.
* UNITS  - number of single family units on a link.
* MULTI  - number of multi family units on a link.
* AREA   - drainage feeding a link.
* LROAD  - the length of road on a link (not including intersection).
* TROAD  - type of road (local or collector).
* WWALK  - sidewalk width.
* BARIER - length of barrier curb.
* ROLLED - length of rolled curb.
* BLVD   - length of boulevard curb.
* NWMLNK - number of watermain links.
* WLINKS - links involved in watermain system.
* NWMGRP - number of tributary groups in watermain system.
* TBTRYW - watermain system 'to'/'from' link associations.
* NDALNK - number of domestic sewer links.
* DLINKS - links involved in domestic sewer system.
* NDSGRP - number of tributary groups in domestic sewer system.
* TBTRYD - domestic sewer system 'to'/'from' link associations.
* NSSLNK - number of storm sewer links.
* SLINKS - links involved in storm sewer system.
* NSSGRP - number of tributary groups in storm sewer system.
* TBTRYS - storm sewer system 'to'/'from' link associations.
* IQ     - number of initial conditions.
* L      - link number of link with initial conditions.
* P      - initial amount of people on a watermain link.
* PDS    - initial amount of people on a domestic sewer link.
* ASS    - initial amount of area for a storm sewer link.
* APARK  - area devoted to park land.
* ABUFF  - area devoted to buffer zones.
* COST*  - various cost constants.
```

```
* DSELEV - elevation of domestic sewer at a node.
* SSELEV - elevation of storm sewer at a node.
* DIAWM  - diameter of each watermain link.
* DIADS  - diameter of each domestic sewer link.
* DIASS  - diameter of each storm sewer link.
* TLCOST - total cost of subdivision.
* ERRFLG - flag indication method of outputting error messages.
* RUNIT1 - read unit #1.
* RUNIT2 - read unit #2.
* WUNIT1 - write unit #1.
* WUNIT1 - write unit #2.
* PEOPLE - average number of people per unit.
* WVMIN  - minimum watermain water velocity (m/s).
* FRICTN - Manning's roughness coefficient (unitless).
* USE    - expected domestic water use per capita (cu. meters/person/day).
* FLOAD  - safety factor for water requirement calculations (unitless).
* DSCOVR - mimimum cover depth over domestic sewer pipes (m).
* DSVMIN - minimum domestic sewer velocity (m/s).
* DSVMAX - maximum domestic sewer velocity (m/s).
* SSCOVR - mimimum cover depth over storm sewer pipes (m).
* SSVMIN - minimum storm sewer velocity (m/s).
* SSVMAX - maximum storm sewer velocity (m/s).
* SPAMAN - mimimum spacing between manholes (m).
* SPALIT - maximum spacing between street lights (m).

      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  NINT,LIFE,NNODE,NTYPE,NLINK,NODE1,NODE2,TROAD,
     &  NWMLNK,WLINKS,NWMGRP,TBTRYW,
     &  NDSLNK,DLINKS,NDSGRP,TBTRYD,
     &  NSSLNK,SLINKS,NSSGRP,TBTRYS,
     &  IQ,L,ERRFLG,RUNIT1,RUNIT2,WUNIT1,WUNIT2,
     &     CBASIN,DINDX2,DINDX3,CONST,
     &  LOPNUM,LOOPS,SNTYPE(100),RESULT
     &     ICDFLG,IDAFLG,TRIFLG,CCTFLG,OPRFLG,MNTFLG,ECNFLG,ACCFLG

      REAL
     &  INTRST,ELEV,LLINK,UNITS,MULTI,AREA,LROAD,WROAD,
     &  WWALK,BARIER,ROLLED,BLVD,P,PDS,ASS,
     &  APARK,ABUFF,RK4,DSELEV,SSELEV,DIAWM,DIADS,DIASS,
     &  CWM,CDS,CSS,TLCOST,PEOPLE,WVMIN,FRICTN,USE,
     &  FLOAD,DSCOVR,DSVMIN,DSVMAX,SSCOVR,SSVMIN,SSVMAX,
     &  SPAMAN,SPALIT,LENGTH,MHCOST,LSCOST,KM,LMCOST,SELEV(100),
     &  LWALK,PIPIDX

      COMMON
     &     /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
     &     /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
     &     /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &     UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &     /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &     /AREA5/LWALK(LINNUM),WWALK(LINNUM),BARIER(LINNUM),
     &  ROLLED(LINNUM),BLVD(LINNUM)
     &     /AREA6/NWMLNK,WLINKS(LINNUM),NWMGRP,TBTRYW(LINNUM,4)
     &     /AREA7/NDSLNK,DLINKS(LINNUM),NDSGRP,TBTRYD(LINNUM,4)
```

```
     &      /AREA8/NSSLNK,SLINKS(LINNUM),NSSGRP,TBTRYS(LINNUM,4)
     &      /AREA9/IQ,L,P(LINNUM),PDS(LINNUM),ASS(LINNUM)
     &      /AREA10/APARK,ABUFF
     &      /AREA11/COST1,COST2
     &      /AREA12/COST3,COST4
     &      /AREA13/COST5,COST6
     &      /AREA14/COST7,COST8,COST9,COST10
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA16/COST19,COST20,COST21,COST22,COST23,COST24
     &      /AREA17/COST25,COST26,COST27,COST28,COST29,COST30,COST31,
     &      COST32,COST33
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
     &      /AREA19/TLCOST
     &      /AREA20/ERRFLG,ICDFLG,IDAFLG,TRIFLG,CCTFLG,OPRFLG,MNTFLG,ECNFLG,
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA22/PEOPLE,WVMIN,FRICTN,USE,FLOAD
     &      /AREA23/DSCOVR,DSVMIN,DSVMAX
     &      /AREA24/SSCOVR,SSVMIN,SSVMAX
     &      /AREA25/SPAMAN,SPALIT
     &   /AREA36/LOPNUM,LOOPS(25,50,2),FLOW(150),FLODIR(150),
     &   HEDLOS(150),PIPDIA(150),HEADS(100),PSEHED(25),PIPIDX(10)
     &      /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
     &      ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
     &      /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
     &      MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
     &      WKCOST,BRCOST,RLCOST,BLCOST
     &      /AREA47/NACCNT,ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100),
     &      DEFINE(100)


      DATA DSELEV/100*9999/SSELEV/100*9999/
      RUNIT1=4
      RUNIT2=7
      WUNIT1=6
      WUNIT2=10

*     ******
*     * The following flags are used to determine which data will be output
*     * With the exception of ERRFLG, the remaining flags will halt output
*     * of their respective data if set to 1.  ERRFLG's use is described in
*     * the variable descriptions.
*     ******

*     * Error-message flag.
      ERRFLG=1

*     * Initial-cost-data flag.
      ICDFLG=0

*     * Initial-data flag.
      IDAFLG=0

*     * Tributary-data flag.
      TRIFLG=0
```

```
*     * Construction-costs flag.
   CCTFLG=0

*     * Operations flag.
   OPRFLG=0

*     * Maintenance flag.
   MNTFLG=0

*     * Economic-analysis flag.
   ECNFLG=0

*     * Accounts-description flag.
   ACCFLG=0

*  THIS INITIATES THE CALCULATION OF THE CONSTRUCTION COSTS
*     * Total construction costs
   TLCOST=0.0

   CALL HEAD

   CALL CLEAR

*  OPEN THE ERROR CHANNEL
   OPEN ( UNIT=WUNIT2, FILE='ERRORFILE', STATUS='NEW' )

   IST=2

   CALL IOFILE(WUNIT1,IST)

   CALL INDAT1(NACCNT)
   IF(ICDFLG.EQ.0)THEN
    CALL OUTDAT1
   ENDIF

   CALL CLEAR

   CALL INDAT2

   IF(IDAFLG.EQ.0)THEN
    CALL OUTDAT2
   ENDIF

   CALL CLEAR

   WRITE (WUNIT1,67)
   WRITE (WUNIT1,68)'C A L I B R A T I O N   D A T A   V A L U E S'
   WRITE (WUNIT1,68)'-----------------------------------------------'
   PRINT*,'C A L I B R A T I O N   D A T A   V A L U E S'
   PRINT*,'-----------------------------------------------'

*****************************************************************
*                                                               *
*                        DEFAULT  VALUES                        *
*                                                               *
*****************************************************************
```

```
      IF ( PEOPLE.EQ.0.0 )     THEN
            PEOPLE=3.35
            WRITE(5,*)'Estimated no. of people per unit     ---
        & defaulted to:',PEOPLE
            WRITE(WUNIT1,68)'Estimated no. of people living in each
        &unit  ---  defaulted to',PEOPLE
      ENDIF
      IF ( DSVMIN.EQ.0.0 )     THEN
            DSVMIN=0.610
            WRITE(5,*)'Min. DOMESTIC SEWER pipe velocity    ---
        & defaulted to:',DSVMIN
            WRITE(WUNIT1,68)'The min. DOMESTIC SEWER velocity within
        &pipe  ---  defaulted to',DSVMIN
      ENDIF
      IF ( SSVMIN.EQ.0.0 )     THEN
            SSVMIN=0.610
            WRITE(5,*)'Min. STORM SEWER pipe velocity       ---
        & defaulted to:',SSVMIN
            WRITE(WUNIT1,68)'The min. STORM SEWER velocity within
        &pipe  ---  defaulted to',SSVMIN
      ENDIF
      IF ( DSVMAX.EQ.0.0 )     THEN
            DSVMAX=2.750
            WRITE(5,*)'Max. DOMESTIC SEWER pipe velocity    ---
        & defaulted to:',DSVMAX
            WRITE(WUNIT1,68)'The max. DOMESTIC SEWER velocity within
        &pipe  ---  defaulted to',DSVMAX
      ENDIF
      IF ( SSVMAX.EQ.0.0 )     THEN
            SSVMAX=3.650
            WRITE(5,*)'Max. STORM SEWER pipe velocity       ---
        & defaulted to:',SSVMAX
            WRITE(WUNIT1,68)'The max. STORM SEWER velocity within
        &pipe  ---  defaulted to',SSVMAX
      ENDIF
      IF ( WVMIN.EQ.0.0 )      THEN
            WVMIN=1.750
            WRITE(5,*)'Min. WATER velocity within pipe      ---
        & defaulted to:',WVMIN
            WRITE(WUNIT1,68)'The min. WATER velocity within
        &pipe  ---  defaulted to',WVMIN
      ENDIF
      IF ( FRICTN.EQ.0.0 )     THEN
            FRICTN=0.013
            WRITE(5,*)'MANNING''s roughness coefficient      ---
        & defaulted to:',FRICTN
            WRITE(WUNIT1,68)'MANNING''s roughness coefficient of the
        &pipe  ---  defaulted to',FRICTN
      ENDIF
      IF ( DSCOVR.EQ.0.0 )     THEN
            DSCOVR=2.750
            WRITE(5,*)'Min. cover depth for DOMESTIC SEWER ---
        & defaulted to:',DSCOVR
            WRITE(WUNIT1,68)'Min. depth of cover for DOMESTIC SEWER
        & ---  defaulted to',DSCOVR
      ENDIF
      IF ( SSCOVR.EQ.0.0 )     THEN
```

```
          SSCOVR=3.350
          WRITE(5,*)'Min. cover depth for STORM SEWER    ---
     & defaulted to:',SSCOVR
          WRITE(WUNIT1,68)'Min. depth of cover for STORM SEWER
     &  ---  defaulted to',SSCOVR
     ENDIF
     IF ( SPAMAN.EQ.0.0 )    THEN
          SPAMAN=100.0
          WRITE(5,*)'Max. spacing between manholes       ---
     & defaulted to:',SPAMAN
          WRITE(WUNIT1,68)'Max. spacing between manholes  ---
     & defaulted to',SPAMAN
     ENDIF
     IF ( SPALIT.EQ.0.0 )    THEN
          SPALIT=70.0
          WRITE(5,*)'Max. spacing between lights         ---
     & defaulted to:',SPALIT
          WRITE(WUNIT1,68)'Max. spacing between lights  ---
     & defaulted to',SPALIT
     ENDIF
     IF ( USE.EQ.0.0 )       THEN
          USE=0.5678
*    Note: This value is the metric equivalent of 150 gal/person/day

          WRITE(5,*)'WATERMAIN domestic water use        ---
     & defaulted to:',USE
          WRITE(WUNIT1,68)'Expected domestic water use for WATERMAIN
     & ---  defaulted to',USE
     ENDIF
     IF ( FLOAD.EQ.0.0 )     THEN
          FLOAD=2.00
          WRITE(5,*)'Water consumption safety factor     ---
     & defaulted to:',FLOAD
          WRITE(WUNIT1,68)'Safety factor for water consumption
     &requirements  ---  defaulted to',FLOAD
     ENDIF
     WRITE (WUNIT1,67)

     PRINT*
     PRINT*,'PROCESSING..................'

     CALL ERRCHK

     CALL IDWATR

*    * Save ELEV for later recall.
     DO 10, L=1, 100
     SELEV(L)=ELEV(L)
10   CONTINUE

     CALL IDDMSW

     CALL IDSTSW

     CALL CREATE

*    * Restore ELEV.
```

```
      DO 20, L=1, 100
      ELEV(L)=SELEV(L)
   20 CONTINUE

*       * Organize nodes:  higher nodes in NODE1 and lower nodes in NODE2.

      DO 30, L=1, NLINK
      IF(ELEV(NODE2(L)).GT.ELEV(NODE1(L)))THEN
      X=NODE1(L)
      NODE1(L)=NODE2(L)
      NODE2(L)=X
      ENDIF
   30 CONTINUE

*       * Eliminate negative values from NTYPE (negative values
*       * indicate an exit node in the CREATE subroutine, but
*       * the rest of RACAM doesn't need the negative values).
      DO 40, L=1, NNODE
      NTYPE(L)=ABS(NTYPE(L))
   40 CONTINUE

      CALL ATTACH

      CALL WATER(WMCOST)

      TLCOST=TLCOST+WMCOST

      CALL DMSTC(DSCOST)

      TLCOST=TLCOST+DSCOST

      CALL STORM(SSCOST)

*     TLCOST=TLCOST+SSCOST

      CALL PAVEMENT

      CALL MANHOLE

      CALL LIGHT

      CALL CATBAS

      CALL SIDEWALK

      CALL SIGN

      CALL UTILITY

      CALL PARKBUFF

      CALL FRONTAGE

      IF(CCTFLG.EQ.0)THEN
       CALL ANALYSIS
      ENDIF
```

```
      CALL OMINIT

      CALL MAINCE

      CALL OPERAT

      IF(ECNFLG.EQ.0)THEN
       CALL ECONOM
      ENDIF

      IF(ACCFLG.EQ.0)THEN
       CALL ACCOUNT

       CALL ACCOUNT1

       CALL ACCOUNT2

       CALL ACCSUM
      ENDIF

      CLOSE (WUNIT1)

      CLOSE (WUNIT2)

   67 FORMAT ('1')
   68 FORMAT (' ',T4,A75,T84,F7.3,/)

      END
```

B.2 SUBROUTINES CALLED BY MAIN ROUTINE

B.2.1 SUBROUTINE: ACCOUNT

```
      SUBROUTINE ACCOUNT


      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &       TIME,WUNIT1

      REAL
        &       MHCOST,LTCOST,INTRST

      CHARACTER*13 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
        &       /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
        &       /AREA10/APARK,ABUFF,PCOST,BCOST
        &       /AREA19/TLCOST
        &       /AREA20/ERRFLG
        &       /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &       /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
        &       MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
        &       WKCOST,BRCOST,RLCOST,BLCOST
        &       /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &       ,MNPACC(25,10),OPPACC(25,10)
        &       /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &       ,DEFINE(100),NACCNT
        &       /AREA48/TCPAVE,GAS1,POW1,PH1,GAS2,POW2,PH2
        &       /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &       ,MNCOST(25,3)
        &       /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &       ,OPCOST(25,3)


      TIME=3

      WRITE (WUNIT1,400)
      DO 100 I=1,TIME
              WRITE (WUNIT1,410)
100           CONTINUE
      WRITE (WUNIT1,420)
      DO 110 I=1,TIME
              WRITE (WUNIT1,430)
110           CONTINUE
      WRITE (WUNIT1,770) INTRST(NINT)
      WRITE (WUNIT1,780) LIFE(NINT)
      WRITE (WUNIT1,450)
      DO 113 I=1,TIME
              WRITE (WUNIT1,455)
113           CONTINUE
      WRITE (WUNIT1,470)
      CALL ACCOUT(1,WMCOST)
```

```
      WRITE (WUNIT1,480)
      CALL ACCOUT(2,DSCOST)
      WRITE (WUNIT1,490)
      CALL ACCOUT(3,SSCOST)
      WRITE (WUNIT1,500)
      CALL ACCOUT(4,TCPAVE)
      WRITE (WUNIT1,510)
      CALL ACCOUT(5,WKCOST)
      WRITE (WUNIT1,520)
      CALL ACCOUT(6,BRCOST)
      WRITE (WUNIT1,530)
      CALL ACCOUT(7,RLCOST)
      WRITE (WUNIT1,540)
      CALL ACCOUT(8,BLCOST)
      WRITE (WUNIT1,550)
      CALL ACCOUT(WUNIT1,PCOST)
      WRITE (WUNIT1,560)
      CALL ACCOUT(10,BCOST)
      WRITE (WUNIT1,570)
      CALL ACCOUT(11,CBCOST)
      WRITE (WUNIT1,600)
      CALL ACCOUT(12,MHCOST)
      WRITE (WUNIT1,610)
      CALL ACCOUT(13,LTCOST)
      WRITE (WUNIT1,620)
      CALL ACCOUT(14,CSIGN)
      WRITE (WUNIT1,630)
      CALL ACCOUT(15,TRACOS)
      WRITE (WUNIT1,640)
      WRITE (WUNIT1,650)
      CALL ACCOUT(16,GAS1)
      WRITE (WUNIT1,660)
      CALL ACCOUT(17,POW1)
      WRITE (WUNIT1,670)
      CALL ACCOUT(18,PH1)
      WRITE (WUNIT1,680)
      WRITE (WUNIT1,650)
      CALL ACCOUT(19,GAS2)
      WRITE (WUNIT1,660)
      CALL ACCOUT(20,POW2)
      WRITE (WUNIT1,670)
      CALL ACCOUT(21,PH2)
      AWORTH=UNFORM(INTRST(NINT),LIFE(NINT),TLCOST)
      WRITE (WUNIT1,750) TLCOST,AWORTH
      DO 114 I=1,TIME
              WRITE (WUNIT1,760) TLCOST,AWORTH
114           CONTINUE
*
*
*
400           FORMAT ('1',' C O N S T R U C T I O N   C O S T
     &B R E A K D O W N')
410           FORMAT ('+',' C O N S T R U C T I O N   C O S T
     &B R E A K D O W N')
420           FORMAT (1X,' """"""""""""""""""""""""""""""""""""""""
     &""""""""""""""""""')
430           FORMAT ('+',' """"""""""""""""""""""""""""""""""""""""
```

```
     &*******************')
*  25,20,20,20
450         FORMAT (/////,6X,'DESCRIPTION',15X,'ACCOUNT NUMBER'
     &,8X,'PRESENT COST',9X,'ANNUAL COST')
455         FORMAT ('+',5X,'DESCRIPTION',15X,'ACCOUNT NUMBER'
     &,8X,'PRESENT COST',9X,'ANNUAL COST')
470         FORMAT (//,6X,'WATERMAIN')
480         FORMAT (6X,'DOMESTIC SEWER')
490         FORMAT (6X,'STORM SEWER')
500         FORMAT (6X,'PAVEMENT')
510         FORMAT (6X,'SIDEWALK')
520         FORMAT (6X,'BARRIER CURB & GUTTER')
530         FORMAT (6X,'ROLLED CURB & GUTTER')
540         FORMAT (6X,'BLVD. CURB & GUTTER')
550         FORMAT (6X,'PARK DEVELOPMENT')
560         FORMAT (6X,'BUFFER DEVELOPMENT')
570         FORMAT (6X,'CATCH-BASIN')
600         FORMAT (6X,'MANHOLE')
610         FORMAT (6X,'STREET LIGHT')
620         FORMAT (6X,'STREET SIGN')
630         FORMAT (6X,'TRAFFIC SIGN')
640         FORMAT (6X,'SINGLE FAMILY')
650         FORMAT (10X,'- GAS')
660         FORMAT (10X,'- POWER')
670         FORMAT (10X,'- TELEPHONE')
680         FORMAT (6X,'MULTI-FAMILY')
750         FORMAT (//,19X,'T   O   T   A   L',17X,2('$',2X,F10.2,7X))
760         FORMAT ('+',18X,'T   O   T   A   L',17X,2('$',2X,F10.2,7X))
770         FORMAT (//,' INTEREST RATE ',5X,': ',F10.2,' %')
780         FORMAT (' SUBDIVISION's LIFE : ',I7,1X,'   yrs.')
*
*
   END
```

B.2.2 SUBROUTINE: ACCOUNT1

```
      SUBROUTINE ACCOUNT1


      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &      TIME,WUNIT1

      REAL IMCOST,MGRAD,MGRATE,MNCOST,MINST,MNPACC

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
     &      ,MNPACC(25,10),OPPACC(25,10)
     &      /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
     &      ,DEFINE(100),NACCNT
     &      /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
     &      ,MNCOST(25,3)
     &      /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
     &      ,OPCOST(25,3)


      TIME=3
*
      WRITE (WUNIT1,400)
      DO 100 I=1,TIME
             WRITE (WUNIT1,410)
100          CONTINUE
      WRITE (WUNIT1,420)
      DO 110 I=1,TIME
             WRITE (WUNIT1,430)
110          CONTINUE
      WRITE (WUNIT1,450)
      DO 113 I=1,TIME
             WRITE (WUNIT1,455)
113          CONTINUE
      WRITE (WUNIT1,470)
      WRITE (WUNIT1,770) MINST(1)
      WRITE (WUNIT1,780) MLIFE(1)
      CALL ACCOUT1(1)
      WRITE (WUNIT1,790) (MNCOST(1,J),J=1,2)
      DO 120 I=1,TIME
             WRITE (WUNIT1,800) (MNCOST(1,J),J=1,2)
120          CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,480)
      WRITE (WUNIT1,770) MINST(2)
      WRITE (WUNIT1,780) MLIFE(2)
      CALL ACCOUT1(2)
      WRITE (WUNIT1,790) (MNCOST(2,J),J=1,2)
      DO 121 I=1,TIME
```

```
            WRITE (WUNIT1,800) (MNCOST(2,J),J=1,2)
121         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,490)
   WRITE (WUNIT1,770) MINST(3)
   WRITE (WUNIT1,780) MLIFE(3)
   CALL ACCOUT1(3)
   WRITE (WUNIT1,790) (MNCOST(3,J),J=1,2)
   DO 122 I=1,TIME
            WRITE (WUNIT1,800) (MNCOST(3,J),J=1,2)
122         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,500)
   WRITE (WUNIT1,770) MINST(4)
   WRITE (WUNIT1,780) MLIFE(4)
   CALL ACCOUT1(4)
   WRITE (WUNIT1,790) (MNCOST(4,J),J=1,2)
   DO 123 I=1,TIME
            WRITE (WUNIT1,800) (MNCOST(4,J),J=1,2)
123         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,510)
   WRITE (WUNIT1,770) MINST(5)
   WRITE (WUNIT1,780) MLIFE(5)
   CALL ACCOUT1(5)
   WRITE (WUNIT1,790) (MNCOST(5,J),J=1,2)
   DO 124 I=1,TIME
            WRITE (WUNIT1,800) (MNCOST(5,J),J=1,2)
124         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,520)
   WRITE (WUNIT1,770) MINST(6)
   WRITE (WUNIT1,780) MLIFE(6)
   CALL ACCOUT1(6)
   WRITE (WUNIT1,790) (MNCOST(6,J),J=1,2)
   DO 125 I=1,TIME
            WRITE (WUNIT1,800) (MNCOST(6,J),J=1,2)
125         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,530)
   WRITE (WUNIT1,770) MINST(7)
   WRITE (WUNIT1,780) MLIFE(7)
   CALL ACCOUT1(7)
   WRITE (WUNIT1,790) (MNCOST(7,J),J=1,2)
   DO 126 I=1,TIME
            WRITE (WUNIT1,800) (MNCOST(7,J),J=1,2)
126         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,540)
   WRITE (WUNIT1,770) MINST(8)
   WRITE (WUNIT1,780) MLIFE(8)
   CALL ACCOUT1(8)
   WRITE (WUNIT1,790) (MNCOST(8,J),J=1,2)
   DO 127 I=1,TIME
            WRITE (WUNIT1,800) (MNCOST(8,J),J=1,2)
127         CONTINUE
   WRITE (WUNIT1,710)
```

```
      WRITE (WUNIT1,550)
      WRITE (WUNIT1,770) MINST(9)
      WRITE (WUNIT1,780) MLIFE(9)
      CALL ACCOUT1(9)
      WRITE (WUNIT1,790) (MNCOST(9,J),J=1,2)
      DO 128 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(9,J),J=1,2)
128           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,560)
      WRITE (WUNIT1,770) MINST(10)
      WRITE (WUNIT1,780) MLIFE(10)
      CALL ACCOUT1(10)
      WRITE (WUNIT1,790) (MNCOST(10,J),J=1,2)
      DO 129 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(10,J),J=1,2)
129           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,570)
      WRITE (WUNIT1,770) MINST(11)
      WRITE (WUNIT1,780) MLIFE(11)
      CALL ACCOUT1(11)
      WRITE (WUNIT1,790) (MNCOST(11,J),J=1,2)
      DO 130 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(11,J),J=1,2)
130           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,600)
      WRITE (WUNIT1,770) MINST(12)
      WRITE (WUNIT1,780) MLIFE(12)
      CALL ACCOUT1(12)
      WRITE (WUNIT1,790) (MNCOST(12,J),J=1,2)
      DO 131 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(12,J),J=1,2)
131           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,610)
      WRITE (WUNIT1,770) MINST(13)
      WRITE (WUNIT1,780) MLIFE(13)
      CALL ACCOUT1(13)
      WRITE (WUNIT1,790) (MNCOST(13,J),J=1,2)
      DO 132 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(13,J),J=1,2)
132           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,620)
      WRITE (WUNIT1,770) MINST(14)
      WRITE (WUNIT1,780) MLIFE(14)
      CALL ACCOUT1(14)
      WRITE (WUNIT1,790) (MNCOST(14,J),J=1,2)
      DO 133 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(14,J),J=1,2)
133           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,630)
      WRITE (WUNIT1,770) MINST(15)
      WRITE (WUNIT1,780) MLIFE(15)
```

```
      CALL ACCOUT1(15)
      WRITE (WUNIT1,790) (MNCOST(15,J),J=1,2)
      DO 134 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(15,J),J=1,2)
134           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,640)
      WRITE (WUNIT1,650)
      WRITE (WUNIT1,770) MINST(16)
      WRITE (WUNIT1,780) MLIFE(16)
      CALL ACCOUT1(16)
      WRITE (WUNIT1,790) (MNCOST(16,J),J=1,2)
      DO 135 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(16,J),J=1,2)
135           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,660)
      WRITE (WUNIT1,770) MINST(17)
      WRITE (WUNIT1,780) MLIFE(17)
      CALL ACCOUT1(17)
      WRITE (WUNIT1,790) (MNCOST(17,J),J=1,2)
      DO 136 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(17,J),J=1,2)
136           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,670)
      WRITE (WUNIT1,770) MINST(18)
      WRITE (WUNIT1,780) MLIFE(18)
      CALL ACCOUT1(18)
      WRITE (WUNIT1,790) (MNCOST(18,J),J=1,2)
      DO 137 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(18,J),J=1,2)
137           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,680)
      WRITE (WUNIT1,650)
      WRITE (WUNIT1,770) MINST(19)
      WRITE (WUNIT1,780) MLIFE(19)
      CALL ACCOUT1(19)
      WRITE (WUNIT1,790) (MNCOST(19,J),J=1,2)
      DO 138 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(19,J),J=1,2)
138           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,660)
      WRITE (WUNIT1,770) MINST(20)
      WRITE (WUNIT1,780) MLIFE(20)
      CALL ACCOUT1(20)
      WRITE (WUNIT1,790) (MNCOST(20,J),J=1,2)
      DO 139 I=1,TIME
              WRITE (WUNIT1,800) (MNCOST(20,J),J=1,2)
139           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,670)
      WRITE (WUNIT1,770) MINST(21)
      WRITE (WUNIT1,780) MLIFE(21)
      CALL ACCOUT1(21)
```

```
         WRITE (WUNIT1,790) (MNCOST(21,J),J=1,2)
         DO 140 I=1,TIME
                 WRITE (WUNIT1,800) (MNCOST(21,J),J=1,2)
      140        CONTINUE
         WRITE (WUNIT1,710)
      *
      *
      *
      400        FORMAT ('1',' M A I N T E N A N C E   C O S T   B R E A K D O W
      410        FORMAT ('+',' M A I N T E N A N C E   C O S T   B R E A K D O W
      420        FORMAT (1X,' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      430        FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
      450        FORMAT (/////,6X,'DESCRIPTION',35X,'ACCOUNT NUMBER'
          &      ,8X,'PRESENT COST',9X,'ANNUAL COST')
      455        FORMAT ('+',5X,'DESCRIPTION',35X,'ACCOUNT NUMBER'
          &      ,8X,'PRESENT COST',9X,'ANNUAL COST')
      470        FORMAT (//,6X,'WATERMAIN')
      480        FORMAT (6X,'DOMESTIC SEWER')
      490        FORMAT (6X,'STORM SEWER')
      500        FORMAT (6X,'PAVEMENT')
      510        FORMAT (6X,'SIDEWALK')
      520        FORMAT (6X,'BARRIER CURB & GUTTER')
      530        FORMAT (6X,'ROLLED CURB & GUTTER')
      540        FORMAT (6X,'BLVD. CURB & GUTTER')
      550        FORMAT (6X,'PARK DEVELOPMENT')
      560        FORMAT (6X,'BUFFER DEVELOPMENT')
      570        FORMAT (6X,'CATCH-BASIN')
      600        FORMAT (6X,'MANHOLE')
      610        FORMAT (6X,'STREET LIGHT')
      620        FORMAT (6X,'STREET SIGN')
      630        FORMAT (6X,'TRAFFIC SIGN')
      640        FORMAT (6X,'SINGLE FAMILY')
      650        FORMAT (10X,'- GAS')
      660        FORMAT (10X,'- POWER')
      670        FORMAT (10X,'- TELEPHONE')
      680        FORMAT (6X,'MULTI-FAMILY')
      710        FORMAT (/)
      770        FORMAT (8X,'INTEREST RATE : ',F10.2,' %')
      780        FORMAT (8X,'SERVICE LIFE  : ',I7,1X,'   yrs.')
      790        FORMAT (21X,'TOTAL',40X,2(7X,'$',2X,F10.2))
      800        FORMAT ('+',20X,'TOTAL',40X,2(7X,'$',2X,F10.2))
      *
      *
         END
```

B.2.3 SUBROUTINE: ACCOUNT2

```
      SUBROUTINE ACCOUNT2


      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &      TIME,OLIFE,WUNIT1

      REAL IOCOST

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &      ,MNPACC(25,10),OPPACC(25,10)
        &      /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &      ,DEFINE(100),NACCNT
        &      /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &      ,MNCOST(25,3)
        &      /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &      ,OPCOST(25,3)


      TIME=3
*
      WRITE (WUNIT1,400)
      DO 100 I=1,TIME
             WRITE (WUNIT1,410)
100          CONTINUE
      WRITE (WUNIT1,420)
      DO 110 I=1,TIME
             WRITE (WUNIT1,430)
110          CONTINUE
      WRITE (WUNIT1,450)
      DO 113 I=1,TIME
             WRITE (WUNIT1,455)
113          CONTINUE
      WRITE (WUNIT1,470)
      WRITE (WUNIT1,770) OINST(1)
      WRITE (WUNIT1,780) OLIFE(1)
      CALL ACCOUT2(1)
      WRITE (WUNIT1,790) (OPCOST(1,J),J=1,2)
      DO 120 I=1,TIME
             WRITE (WUNIT1,800) (OPCOST(1,J),J=1,2)
120          CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,480)
      WRITE (WUNIT1,770) OINST(2)
      WRITE (WUNIT1,780) OLIFE(2)
      CALL ACCOUT2(2)
      WRITE (WUNIT1,790) (OPCOST(2,J),J=1,2)
      DO 121 I=1,TIME
```

```
            WRITE (WUNIT1,800) (OPCOST(2,J),J=1,2)
121         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,490)
   WRITE (WUNIT1,770) OINST(3)
   WRITE (WUNIT1,780) OLIFE(3)
   CALL ACCOUT2(3)
   WRITE (WUNIT1,790) (OPCOST(3,J),J=1,2)
   DO 122 I=1,TIME
            WRITE (WUNIT1,800) (OPCOST(3,J),J=1,2)
122         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,500)
   WRITE (WUNIT1,770) OINST(4)
   WRITE (WUNIT1,780) OLIFE(4)
   CALL ACCOUT2(4)
   WRITE (WUNIT1,790) (OPCOST(4,J),J=1,2)
   DO 123 I=1,TIME
            WRITE (WUNIT1,800) (OPCOST(4,J),J=1,2)
123         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,510)
   WRITE (WUNIT1,770) OINST(5)
   WRITE (WUNIT1,780) OLIFE(5)
   CALL ACCOUT2(5)
   WRITE (WUNIT1,790) (OPCOST(5,J),J=1,2)
   DO 124 I=1,TIME
            WRITE (WUNIT1,800) (OPCOST(5,J),J=1,2)
124         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,520)
   WRITE (WUNIT1,770) OINST(6)
   WRITE (WUNIT1,780) OLIFE(6)
   CALL ACCOUT2(6)
   WRITE (WUNIT1,790) (OPCOST(6,J),J=1,2)
   DO 125 I=1,TIME
            WRITE (WUNIT1,800) (OPCOST(6,J),J=1,2)
125         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,530)
   WRITE (WUNIT1,770) OINST(7)
   WRITE (WUNIT1,780) OLIFE(7)
   CALL ACCOUT2(7)
   WRITE (WUNIT1,790) (OPCOST(7,J),J=1,2)
   DO 126 I=1,TIME
            WRITE (WUNIT1,800) (OPCOST(7,J),J=1,2)
126         CONTINUE
   WRITE (WUNIT1,710)
   WRITE (WUNIT1,540)
   WRITE (WUNIT1,770) OINST(8)
   WRITE (WUNIT1,780) OLIFE(8)
   CALL ACCOUT2(8)
   WRITE (WUNIT1,790) (OPCOST(8,J),J=1,2)
   DO 127 I=1,TIME
            WRITE (WUNIT1,800) (OPCOST(8,J),J=1,2)
127         CONTINUE
   WRITE (WUNIT1,710)
```

```
      WRITE (WUNIT1,550)
      WRITE (WUNIT1,770) OINST(9)
      WRITE (WUNIT1,780) OLIFE(9)
      CALL ACCOUT2(9)
      WRITE (WUNIT1,790) (OPCOST(9,J),J=1,2)
      DO 128 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(9,J),J=1,2)
128           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,560)
      WRITE (WUNIT1,770) OINST(10)
      WRITE (WUNIT1,780) OLIFE(10)
      CALL ACCOUT2(10)
      WRITE (WUNIT1,790) (OPCOST(10,J),J=1,2)
      DO 129 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(10,J),J=1,2)
129           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,570)
      WRITE (WUNIT1,770) OINST(11)
      WRITE (WUNIT1,780) OLIFE(11)
      CALL ACCOUT2(11)
      WRITE (WUNIT1,790) (OPCOST(11,J),J=1,2)
      DO 130 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(11,J),J=1,2)
130           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,600)
      WRITE (WUNIT1,770) OINST(12)
      WRITE (WUNIT1,780) OLIFE(12)
      CALL ACCOUT2(12)
      WRITE (WUNIT1,790) (OPCOST(12,J),J=1,2)
      DO 131 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(12,J),J=1,2)
131           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,610)
      WRITE (WUNIT1,770) OINST(13)
      WRITE (WUNIT1,780) OLIFE(13)
      CALL ACCOUT2(13)
      WRITE (WUNIT1,790) (OPCOST(13,J),J=1,2)
      DO 132 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(13,J),J=1,2)
132           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,620)
      WRITE (WUNIT1,770) OINST(14)
      WRITE (WUNIT1,780) OLIFE(14)
      CALL ACCOUT2(14)
      WRITE (WUNIT1,790) (OPCOST(14,J),J=1,2)
      DO 133 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(14,J),J=1,2)
133           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,630)
      WRITE (WUNIT1,770) OINST(15)
      WRITE (WUNIT1,780) OLIFE(15)
```

```
      CALL ACCOUT2(15)
      WRITE (WUNIT1,790) (OPCOST(15,J),J=1,2)
      DO 134 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(15,J),J=1,2)
134           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,640)
      WRITE (WUNIT1,650)
      WRITE (WUNIT1,770) OINST(16)
      WRITE (WUNIT1,780) OLIFE(16)
      CALL ACCOUT2(16)
      WRITE (WUNIT1,790) (OPCOST(16,J),J=1,2)
      DO 135 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(16,J),J=1,2)
135           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,660)
      WRITE (WUNIT1,770) OINST(17)
      WRITE (WUNIT1,780) OLIFE(17)
      CALL ACCOUT2(17)
      WRITE (WUNIT1,790) (OPCOST(17,J),J=1,2)
      DO 136 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(17,J),J=1,2)
136           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,670)
      WRITE (WUNIT1,770) OINST(18)
      WRITE (WUNIT1,780) OLIFE(18)
      CALL ACCOUT2(18)
      WRITE (WUNIT1,790) (OPCOST(18,J),J=1,2)
      DO 137 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(18,J),J=1,2)
137           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,680)
      WRITE (WUNIT1,650)
      WRITE (WUNIT1,770) OINST(19)
      WRITE (WUNIT1,780) OLIFE(19)
      CALL ACCOUT2(19)
      WRITE (WUNIT1,790) (OPCOST(19,J),J=1,2)
      DO 138 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(19,J),J=1,2)
138           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,660)
      WRITE (WUNIT1,770) OINST(20)
      WRITE (WUNIT1,780) OLIFE(20)
      CALL ACCOUT2(20)
      WRITE (WUNIT1,790) (OPCOST(20,J),J=1,2)
      DO 139 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(20,J),J=1,2)
139           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,670)
      WRITE (WUNIT1,770) OINST(21)
      WRITE (WUNIT1,780) OLIFE(21)
      CALL ACCOUT2(21)
```

```
      WRITE (WUNIT1,790) (OPCOST(21,J),J=1,2)
      DO 140 I=1,TIME
              WRITE (WUNIT1,800) (OPCOST(21,J),J=1,2)
140           CONTINUE
      WRITE (WUNIT1,710)
*
*
*
400           FORMAT ('1',' O P E R A T I O N   C O S T   B R E A K D O W N')
410           FORMAT ('+',' O P E R A T I O N   C O S T   B R E A K D O W N')
420           FORMAT (1X,' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
430           FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
450           FORMAT (/////,6X,'DESCRIPTION',35X,'ACCOUNT NUMBER'
     &        ,8X,'PRESENT COST',9X,'ANNUAL COST')
455           FORMAT ('+',5X,'DESCRIPTION',35X,'ACCOUNT NUMBER'
     &        ,8X,'PRESENT COST',9X,'ANNUAL COST')
470           FORMAT (//,6X,'WATERMAIN')
480           FORMAT (6X,'DOMESTIC SEWER')
490           FORMAT (6X,'STORM SEWER')
500           FORMAT (6X,'PAVEMENT')
510           FORMAT (6X,'SIDEWALK')
520           FORMAT (6X,'BARRIER CURB & GUTTER')
530           FORMAT (6X,'ROLLED CURB & GUTTER')
540           FORMAT (6X,'BLVD. CURB & GUTTER')
550           FORMAT (6X,'PARK DEVELOPMENT')
560           FORMAT (6X,'BUFFER DEVELOPMENT')
570           FORMAT (6X,'CATCH-BASIN')
600           FORMAT (6X,'MANHOLE')
610           FORMAT (6X,'STREET LIGHT')
620           FORMAT (6X,'STREET SIGN')
630           FORMAT (6X,'TRAFFIC SIGN')
640           FORMAT (6X,'SINGLE FAMILY')
650           FORMAT (10X,'- GAS')
660           FORMAT (10X,'- POWER')
670           FORMAT (10X,'- TELEPHONE')
680           FORMAT (6X,'MULTI-FAMILY')
710           FORMAT (/)
770           FORMAT (8X,'INTEREST RATE : ',F10.2,' %')
780           FORMAT (8X,'SERVICE LIFE  : ',I7,1X,'   yrs.')
790           FORMAT (21X,'TOTAL',40X,2(7X,'$',2X,F10.2))
800           FORMAT ('+',20X,'TOTAL',40X,2(7X,'$',2X,F10.2))
*
*
      END
```

B.2.4 SUBROUTINE: ACCSUM

```
      SUBROUTINE ACCSUM


      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &       TIME,WUNIT1

      REAL
        &       MHCOST,LTCOST,INTRST

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*35 DEFINE

      COMMON
        &       /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
        &       /AREA10/APARK,ABUFF,PCOST,BCOST
        &       /AREA19/TLCOST,TMCOST,TOCOST
        &       /AREA20/ERRFLG
        &       /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &       /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
        &       MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
        &       WKCOST,BRCOST,RLCOST,BLCOST
        &       /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &       ,MNPACC(25,10),OPPACC(25,10)
        &       /AREA47/NACCNT,ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100
        &       ,DEFINE(100)
        &       /AREA48/TCPAVE,GAS1,POW1,PH1,GAS2,POW2,PH2
        &       /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &       ,MNCOST(25,3)
        &       /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &       ,OPCOST(25,3)


      TIME=3

*
      TOTAL=ACCOST(1)+AMCOST(1)+AOCOST(1)
      WRITE (WUNIT1,690)
      DO 115 I=1,TIME
              WRITE (WUNIT1,695)
115           CONTINUE
      WRITE (WUNIT1,700)
      DO 117 I=1,TIME
              WRITE (WUNIT1,705)
117           CONTINUE
      WRITE (WUNIT1,710)
      WRITE (WUNIT1,720) DEFINE(1),ACCNUM(1),ACCOST(1),AMCOST(1),AOCOST(1)
      DO 120 I=2,NACCNT
              TOTAL=ACCOST(I)+AMCOST(I)+AOCOST(I)
              WRITE (WUNIT1,720) DEFINE(I),ACCNUM(I),ACCOST(I),AMCOST(I),
        &                AOCOST(I)
120           CONTINUE
      WRITE (WUNIT1,730) TLCOST,TMCOST,TOCOST
```

```
      DO 130 I=1,TIME
             WRITE (WUNIT1,740) TLCOST,TMCOST,TOCOST
130          CONTINUE
*
*
*
690          FORMAT ('1','SUMMARY OF ACCOUNTS :')
695          FORMAT ('+','SUMMARY OF ACCOUNTS :')
700          FORMAT (///,22X,'DESCRIPTION',25X,'ACCOUNT NUMBER',3X,
     &'CONSTRUCTION COST',4X,'MAINTENANCE COST',
     &6X,'OPERATION COST')
705          FORMAT ('+',21X,'DESCRIPTION',25X,'ACCOUNT NUMBER',3X,
     &'CONSTRUCTION COST',4X,'MAINTENANCE COST',
     &6X,'OPERATION COST')
710          FORMAT (//)
720          FORMAT (1X,A50,8X,A12,3(7X,'$',2X,F10.2))
730          FORMAT (//,22X,'T O T A L',40X,3(7X,'$',2X,F10.2))
740          FORMAT ('+',21X,'T O T A L',40X,3(7X,'$',2X,F10.2))


      RETURN
      END
```

B.2.5 SUBROUTINE: ANALYSIS

```
      SUBROUTINE ANALYSIS

      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER XSEC,XSPA,COL,XTOTAL,X,SPACE
     &       ,YTOTAL,YSEC,DSMAN,SSMAN,TRAFIC,HLINK
     &       ,EVEN,TIME,TDSMAN,TSSMAN,TSIGN,TROAD,
     &       FLAG,WUNIT1

      REAL
     &       LCBCOS,LTCOSL,LROAD,LLINK,MULTI,MULPGA,
     &       MULPNA,MHCOST,LTCOST,SSCS(LINNUM),DSCS(LINNUM)
     &       ,WMCS(LINNUM),LOCLEN,LSCOST,CNCOSL(LINNUM)

      CHARACTER*132 LINE,LINE1,LINE2,LINE3,LINE4

      COMMON
     &       /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
     &       /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &       RCURB
     &       /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &       UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &       /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &       /AREA5/LWALK(LINNUM),WWALK(LINNUM),BARIER(LINNUM),ROLLED(LINNUM)
     &       ,BLVD(LINNUM)
     &       /AREA9/IQ,L,P(LINNUM),PDS(LINNUM),ASS(LINNUM)
     &       /AREA10/APARK,ABUFF,PCOST,BCOST
     &       /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &       DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
     &       /AREA19/TLCOST
     &       /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &       /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &       ,PVCOST(LINNUM),ACOL,ALOC,COLLEN,LOCLEN
     &       /AREA29/DSMAN(LINNUM),SSMAN(LINNUM),DSCOSL(LINNUM),
     &       SSCOSL(LINNUM),DSCOSN(NODNUM),SSCOSN(NODNUM)
     &       /AREA30/CBLEAD(4,2),CBDPTH(4),FLINK(NODNUM,4),
     &       BASIN(NODNUM),CBCOS(NODNUM),LBASIN(LINNUM),LCBCOS(LINNUM)
     &       ,STRCAT(NODNUM,NODNUM,4)
     &       /AREA31/TLEN,TAREA,RDPINT(LINNUM)
     &       /AREA34/STRLIT(NODNUM,NODNUM,4),LTCOSL(LINNUM),LTCOSN(NODNUM),
     &       LTLINK(LINNUM),LTNODE(NODNUM)
     &       /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
     &       ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
     &       /AREA38/NYELD(NODNUM),NSTOP(NODNUM),TYELD,TSTOP,NSIGN(LINNUM),
     &       LYELD(LINNUM),LSTOP(LINNUM),SGCOST(LINNUM),YDCOST(LINNUM),
     &       STCOST(LINNUM)
     &       /AREA39/LINE,LINE1,LINE2,LINE3,LINE4
     &       /AREA40/TFRONT,GROSS,YSTLEN,YGROSS
     &       /AREA41/NCUL,N3WAY,N4WAY,NEXIT,NDEAD
     &       /AREA42/REVMAX,REVMIN,ASSUME
     &       /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
     &       MHCOST,NBASIN,CBCOST,NTRAFF,TFCOST,TWK,TBR,TRL,TBL,
     &       WKCOST,BRCOST,RLCOST,BLCOST
     &       /AREA44/PWSING(LINNUM),PWMULT(LINNUM),GSSING(LINNUM)
```

```
      &      ,GSMULT(LINNUM),PHSING(LINNUM),PHMULT(LINNUM),UTCOST(LINNUM)
      &      /AREA45/WKCOSL(LINNUM),BRCOSL(LINNUM),RLCOSL(LINNUM)
      &      ,BLCOSL(LINNUM)
      &      /AREA48/TCPAVE,GAS1,POW1,PH1,GAS2,POW2,PH2
      &      /AREA51/TPAREA,SSPIPE,WMPIPE,DSPIPE
*
   YSEC=5
   COL=7
   TIME=4
*
*
*
*  sewer/watermain system
*
*  storm sewer
*
   WRITE (WUNIT1,500)
   DO 100 I=1,TIME
          WRITE (WUNIT1,502)
100       CONTINUE
   CALL SWTABLE(COL,DIASS,CSS,SSCS,SSPIPE)
   WRITE (WUNIT1,410) SSPIPE,SSCOST
   DO 105 I=1,TIME
          WRITE (WUNIT1,423)
105       CONTINUE
   WRITE (WUNIT1,400) LINE
*
*  domestic sewer
*
   WRITE (WUNIT1,504)
   DO 110 I=1,TIME
          WRITE (WUNIT1,506)
110       CONTINUE
   CALL SWTABLE(COL,DIADS,CDS,DSCS,DSPIPE)
   WRITE (WUNIT1,410) DSPIPE,DSCOST
   DO 115 I=1,TIME
          WRITE (WUNIT1,423)
115       CONTINUE
   WRITE (WUNIT1,400) LINE
*
*  watermain
*
   WRITE (WUNIT1,508)
   DO 120 I=1,TIME
          WRITE (WUNIT1,510)
120       CONTINUE
   CALL SWTABLE(COL,DIAWM,CWM,WMCS,WMPIPE)
   WRITE (WUNIT1,410) WMPIPE,WMCOST
   DO 125 I=1,TIME
          WRITE (WUNIT1,423)
125       CONTINUE
   WRITE (WUNIT1,400) LINE
*
*
*
*  manholes/catch-basins table
   XSEC=6
```

```
      XSPA=20
      CALL CLINE(XSPA,XSEC,COL,0,0)
      MCOL=XSPA*XSEC
*
      LENGTH=LEN('DOMESTIC SEWER MANHOLE')
      IFRONT=INT((2*XSPA-LENGTH)/2.0)+COL+2
      LINE3(IFRONT:IFRONT+LENGTH)='DOMESTIC SEWER MANHOLE'
*
      LENGTH=LEN('STORM SEWER MANHOLE')
      IFRONT=INT((2*XSPA-LENGTH)/2.0)+COL+XSPA*2+2
      LINE3(IFRONT:IFRONT+LENGTH)='STORM SEWER MANHOLE'
*
      LENGTH=LEN('STORM SEWER CATCH-BASIN')
      IFRONT=INT((2*XSPA-LENGTH)/2.0)+COL+XSPA*4+2
      LINE3(IFRONT:IFRONT+LENGTH)='STORM SEWER CATCH-BASIN'
*
      LENGTH=LEN('LINK')
      IFRONT=INT((5-LENGTH)/2.0)+2
      LINE4(IFRONT:IFRONT+LENGTH)='LINK'
*
      DO 140 I=1,3
              SPACE=XSPA*2*(I-1)+2
              LENGTH=LEN('QUANTITY')
              IFRONT=INT((XSPA-LENGTH)/2.0)+COL+SPACE
              LINE4(IFRONT:IFRONT+LENGTH)='QUANTITY'
              LENGTH=LEN('COST')
              IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+SPACE
              LINE4(IFRONT:IFRONT+LENGTH)='COST'
140           CONTINUE
*
      WRITE (WUNIT1,520)
      DO 150 I=1,TIME
              WRITE (WUNIT1,530)
150           CONTINUE
      WRITE (WUNIT1,400) LINE
      WRITE (WUNIT1,400) LINE3
      WRITE (WUNIT1,400) LINE2
      WRITE (WUNIT1,400) LINE4
      WRITE (WUNIT1,400) LINE1
*
*     set the total/counter to zero
      TDSMAN=0
      TSSMAN=0
      TSSCAT=0
      TDSCOS=0.0
      TSSCOS=0.0
      TCBCOS=0.0
*
      YTOTAL=YSEC+1
      DO 160 L=1,NLINK
              IF ( L.EQ.YTOTAL )  THEN
                WRITE (WUNIT1,400) LINE1
                YTOTAL=YTOTAL+YSEC
              ENDIF
*
              TDSMAN=TDSMAN+DSMAN(L)
              TDSCOS=TDSCOS+DSCOSL(L)
```

```
            TSSMAN=TSSMAN+SSMAN(L)
            TSSCOS=TSSCOS+SSCOSL(L)
            TSSCAT=TSSCAT+LBASIN(L)
            TCBCOS=TCBCOS+LCBCOS(L)
*
            WRITE(WUNIT1,420) L,DSMAN(L),DSCOSL(L),SSMAN(L),SSCOSL(L),
     &              LBASIN(L),LCBCOS(L)
160         CONTINUE
*
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,421) TDSMAN,TDSCOS,TSSMAN,TSSCOS,NBASIN,CBCOST
   DO 165 I=1,TIME
            WRITE (WUNIT1,423)
165         CONTINUE
   WRITE (WUNIT1,400) LINE
*
*
*
*  light/street length/sign
   XSEC=6
   XSPA=20
   CALL CLINE(XSPA,XSEC,COL,0,0)
   MCOL=XSPA*XSEC

*
   LENGTH=LEN('STREET LIGHT')
   IFRONT=INT((2*XSPA-LENGTH)/2.0)+COL+2
   LINE3(IFRONT:IFRONT+LENGTH)='STREET LIGHT'
*
   LENGTH=LEN('STREET SIGN')
   IFRONT=INT((2*XSPA-LENGTH)/2.0)+COL+XSPA*2+2
   LINE3(IFRONT:IFRONT+LENGTH)='STREET SIGN'
*
   LENGTH=LEN('TRAFFIC SIGN')
   IFRONT=INT((2*XSPA-LENGTH)/2.0)+COL+XSPA*4+2
   LINE3(IFRONT:IFRONT+LENGTH)='TRAFFIC SIGN'
*
   LENGTH=LEN('LINK')
   IFRONT=INT((5-LENGTH)/2.0)+2
   LINE4(IFRONT:IFRONT+LENGTH)='LINK'
*
   DO 170 I=1,3
            SPACE=XSPA*2*(I-1)+2
            LENGTH=LEN('QUANTITY')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='QUANTITY'
            LENGTH=LEN('COST')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='COST'
170         CONTINUE
*
   WRITE (WUNIT1,540)
   DO 180 I=1,TIME
            WRITE (WUNIT1,545)
180         CONTINUE
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,400) LINE3
```

```
      WRITE (WUNIT1,400) LINE2
      WRITE (WUNIT1,400) LINE4
      WRITE (WUNIT1,400) LINE1
*
      YTOTAL=YSEC+1
      DO 183 L=1,NLINK
              IF ( L.EQ.YTOTAL )  THEN
                WRITE (WUNIT1,400) LINE1
                YTOTAL=YTOTAL+YSEC
              ENDIF
              TRAFIC=LSTOP(L)+LYELD(L)
              TRACOS=STCOST(L)+YDCOST(L)
              WRITE(WUNIT1,430) L,LTLINK(L),LTCOSL(L),NSIGN(L),SGCOST(L),
     &                TRAFIC,TRACOS
183           CONTINUE
      WRITE (WUNIT1,400) LINE
      WRITE (WUNIT1,431) NLIGHT,LTCOST,TSIGN,CSIGN,NTRAFF,TFCOST
      DO 185 I=1,TIME
              WRITE (WUNIT1,423)
185           CONTINUE
      WRITE (WUNIT1,400) LINE
*
*
*
*  pavement
      XSEC=9
      XSPA=13
      CALL CLINE(XSPA,XSEC,COL,0,0)
      MCOL=XSPA*XSEC
*
      DO 190 J=1,XSEC-1
              LINE2(COL+J*XSPA:COL+J*XSPA)='-'
              LINE3(COL+J*XSPA:COL+J*XSPA)=' '
              LINE4(COL+J*XSPA:COL+J*XSPA)=' '
190           CONTINUE
*
      LENGTH=LEN('LOCAL STREET')
      IFRONT=INT((3*XSPA-LENGTH)/2.0)+COL+2
      LINE3(IFRONT:IFRONT+LENGTH)='LOCAL STREET'
*
      LENGTH=LEN('COLLECTOR STREET')
      IFRONT=INT((3*XSPA-LENGTH)/2.0)+COL+XSPA*3+2
      LINE3(IFRONT:IFRONT+LENGTH)='COLLECTOR STREET'
*
      LENGTH=LEN('TOTAL')
      IFRONT=INT((3*XSPA-LENGTH)/2.0)+COL+XSPA*6+2
      LINE3(IFRONT:IFRONT+LENGTH)='TOTAL'
*
      LENGTH=LEN('LINK')
      IFRONT=INT((5-LENGTH)/2.0)+2
      LINE4(IFRONT:IFRONT+LENGTH)='LINK'
*
      DO 191 I=1,3
              SPACE=XSPA*3*(I-1)+1
              LENGTH=LEN('LENGTH')
              IFRONT=INT((XSPA-LENGTH)/2.0)+COL+SPACE
              LINE4(IFRONT:IFRONT+LENGTH)='LENGTH'
```

```
            LENGTH=LEN('AREA ( sq.m )')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='AREA ( sq.m )'
            LENGTH=LEN('COST')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*2+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='COST'
191         CONTINUE
*
   LINE2(COL+3*XSPA:COL+3*XSPA)='+'
   LINE2(COL+6*XSPA:COL+6*XSPA)='+'
   LINE3(COL+3*XSPA:COL+3*XSPA)='!'
   LINE3(COL+6*XSPA:COL+6*XSPA)='!'
   LINE4(COL+3*XSPA:COL+3*XSPA)='!'
   LINE4(COL+6*XSPA:COL+6*XSPA)='!'

*
   WRITE (WUNIT1,550)
   DO 192 I=1,TIME
            WRITE (WUNIT1,555)
192         CONTINUE
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,400) LINE3
   WRITE (WUNIT1,400) LINE2
   WRITE (WUNIT1,400) LINE4
   WRITE (WUNIT1,400) LINE1
*
*  set the total to zero
   LSCOST=0.0
   CSCOST=0.0
*
   YTOTAL=YSEC+1
   DO 193 L=1,NLINK
            IF ( L.EQ.YTOTAL )  THEN
              WRITE (WUNIT1,400) LINE1
              YTOTAL=YTOTAL+YSEC
            ENDIF
            IF ( TROAD(L).EQ.1 )  THEN
              CSCOST=CSCOST+PVCOST(L)
              WRITE (WUNIT1,470) L,0.0,0.0,0.0,RDPINT(L),ALINK(L)
     &                ,PVCOST(L),RDPINT(L),ALINK(L),PVCOST(L)
            ELSE
              LSCOST=LSCOST+PVCOST(L)
              WRITE (WUNIT1,470) L,RDPINT(L),ALINK(L),PVCOST(L),0.0
     &                ,0.0,0.0,RDPINT(L),ALINK(L),PVCOST(L)
            ENDIF
193         CONTINUE
   TCPAVE=LSCOST+CSCOST
   CALL ASPLIT(4,TCPAVE)
   TPLEN=COLLEN+LOCLEN
   TPAREA=ACOL+ALOC
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,471) LOCLEN,ALOC,LSCOST,COLLEN,ACOL,CSCOST,
     &      TPLEN,TPAREA,TCPAVE
   DO 194 I=1,TIME
            WRITE (WUNIT1,423)
194         CONTINUE
   WRITE (WUNIT1,400) LINE
```

```
*
*
*
*
*  sidewalk/curb
   XSEC=5
   XSPA=24
   CALL CLINE(XSPA,XSEC,COL,0,0)
   MCOL=XSPA*XSEC
*
   LENGTH=LEN('BARRIER CURB/GUTTER')
   IFRONT=INT((XSPA-LENGTH)/2.0)+COL+2
   LINE3(IFRONT:IFRONT+LENGTH)='BARRIER CURB/GUTTER'
*
   LENGTH=LEN('ROLLED CURB/GUTTER')
   IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+2
   LINE3(IFRONT:IFRONT+LENGTH)='ROLLED CURB/GUTTER'
*
   LENGTH=LEN('BLVD. CURB/GUTTER')
   IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*2+2
   LINE3(IFRONT:IFRONT+LENGTH)='BLVD. CURB/GUTTER'
*
   LENGTH=LEN('SIDEWALK')
   IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*3+2
   LINE3(IFRONT:IFRONT+LENGTH)='SIDEWALK'
*
   LENGTH=LEN('LINK')
   IFRONT=INT((5-LENGTH)/2.0)+2
   LINE4(IFRONT:IFRONT+LENGTH)='LINK'
*
   LINE3(COL+XSPA:COL+XSPA)='!'
   LINE4(COL+XSPA:COL+XSPA)='!'
   LINE3(COL+XSPA*3:COL+XSPA*3)='!'
   LINE4(COL+XSPA*3:COL+XSPA*3)='!'
*
   DO 200 I=1,3
            SPACE=XSPA*(I-1)+2
            LENGTH=LEN('LENGTH')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='LENGTH'
200         CONTINUE
            SPACE=XSPA*3+1
            LENGTH=LEN('AREA (sq. m)')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='AREA (sq. m)'
            LENGTH=LEN('COST')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='COST'

   WRITE (WUNIT1,580)
   DO 210 I=1,TIME
            WRITE (WUNIT1,590)
210         CONTINUE
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,400) LINE3
   WRITE (WUNIT1,400) LINE2
   WRITE (WUNIT1,400) LINE4
```

```
      WRITE (WUNIT1,400) LINE1
*
*     set the total/counter to zero
      TSRCOS=0.0
*
      YTOTAL=YSEC+1
      DO 220 L=1,NLINK
              IF ( L.EQ.YTOTAL )  THEN
                WRITE (WUNIT1,400) LINE1
                YTOTAL=YTOTAL+YSEC
              ENDIF
              TSRCOS=TSRCOS+SRCOST(L)
              WRITE(WUNIT1,460) L,BARIER(L),ROLLED(L),BLVD(L)
     &                ,ASIDE(L),SRCOST(L)
220           CONTINUE
      WRITE (WUNIT1,400) LINE
      WRITE (WUNIT1,461) TBR,TRL,TBL,TWK,TSRCOS
      DO 225 I=1,TIME
              WRITE (WUNIT1,423)
225           CONTINUE
      WRITE (WUNIT1,400) LINE
*
*
*
*     single family/multi-family units
      XSEC=9
      XSPA=13
      CALL CLINE(XSPA,XSEC,COL,0,0)
      MCOL=XSPA*XSEC
*
      LENGTH=LEN('SINGLE FAMILY')
      IFRONT=INT((4*XSPA-LENGTH)/2.0)+COL+2
      LINE3(IFRONT:IFRONT+LENGTH)='SINGLE FAMILY'
*
      LENGTH=LEN('MULTI-FAMILY')
      IFRONT=INT((4*XSPA-LENGTH)/2.0)+COL+XSPA*4+2
      LINE3(IFRONT:IFRONT+LENGTH)='MULTI-FAMILY'
*
      LENGTH=LEN('LINK')
      IFRONT=INT((5-LENGTH)/2.0)+2
      LINE4(IFRONT:IFRONT+LENGTH)='LINK'
*
      LENGTH=LEN('TOTAL')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*8+SPACE
      LINE4(IFRONT:IFRONT+LENGTH)='TOTAL'
*
      DO 230 I=1,2
              SPACE=XSPA*4*(I-1)+1
              LENGTH=LEN('UNITS')
              IFRONT=INT((XSPA-LENGTH)/2.0)+COL+SPACE
              LINE4(IFRONT:IFRONT+LENGTH)='UNITS'
              LENGTH=LEN('GAS')
              IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+SPACE
              LINE4(IFRONT:IFRONT+LENGTH)='GAS'
              LENGTH=LEN('POWER')
              IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*2+SPACE
              LINE4(IFRONT:IFRONT+LENGTH)='POWER'
```

```
            LENGTH=LEN('TELEPHONE')
            IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*3+SPACE
            LINE4(IFRONT:IFRONT+LENGTH)='TELEPHONE'
230         CONTINUE
*
   LENGTH=LEN('TOTAL')
   IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*8+1
   LINE4(IFRONT:IFRONT+LENGTH)='TOTAL'
*
   LINE4(COL+XSPA*2:COL+XSPA*2)=' '
   LINE4(COL+XSPA*6:COL+XSPA*6)=' '
*
   WRITE (WUNIT1,560)
   DO 240 I=1,TIME
            WRITE (WUNIT1,570)
240         CONTINUE
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,400) LINE3
   WRITE (WUNIT1,400) LINE2
   WRITE (WUNIT1,400) LINE4
   WRITE (WUNIT1,400) LINE1
*
*  set the total/counter to zero
   GAS1=0.0
   POW1=0.0
   PH1=0.0
   GAS2=0.0
   POW2=0.0
   PH2=0.0
   UTOTAL=0.0
*
   YTOTAL=YSEC+1
   DO 241 L=1,NLINK
            IF ( L.EQ.YTOTAL )  THEN
              WRITE (WUNIT1,400) LINE1
              YTOTAL=YTOTAL+YSEC
            ENDIF
*
            GAS1=GAS1+GSSING(L)
            POW1=POW1+PWSING(L)
            PH1=PH1+PHSING(L)
            GAS2=GAS2+GSMULT(L)
            POW2=POW2+PWMULT(L)
            PH2=PH2+PHMULT(L)
            UTOTAL=UTOTAL+UTCOST(L)
*
            WRITE (WUNIT1,450) L,INT(UNITS(L)),GSSING(L),PWSING(L),
     &              PHSING(L),INT(MULTI(L)),GSMULT(L),PWMULT(L),
     &              PHMULT(L),UTCOST(L)
241         CONTINUE
   CALL ASPLIT(16,GAS1)
   CALL ASPLIT(17,POW1)
   CALL ASPLIT(18,PH1)
   CALL ASPLIT(19,GAS2)
   CALL ASPLIT(20,POW2)
   CALL ASPLIT(21,PH2)
   TUSING=GAS1+POW1+PH1
```

```
      TUMULT=GAS2+POW2+PH2
      WRITE (WUNIT1,400) LINE
      WRITE (WUNIT1,451) INT(TSING),GAS1,POW1,PH1,
     &      INT(TMULT),GAS2,POW2,PH2,UTOTAL
      DO 245 I=1,TIME
              WRITE (WUNIT1,423)
245           CONTINUE
      WRITE (WUNIT1,400) LINE
*
*
*  construction cost for each link
*
      XSEC=5
      XSPA=24
      CALL CLINE(XSPA,XSEC,COL,0,0)
      MCOL=XSPA*XSEC
*
      TCONST=0.0
*
      DO 246 L=1,NLINK
              VAL1=SSCS(L)+DSCS(L)+WMCS(L)
              VAL2=DSCOSL(L)+SSCOSL(L)+LCBCOS(L)
              VAL3=STCOST(L)+YDCOST(L)+SGCOST(L)
              VAL4=LTCOSL(L)+PVCOST(L)
              VAL5=GSSING(L)+PWSING(L)+PHSING(L)
              VAL6=GSMULT(L)+PWMULT(L)+PHMULT(L)
              VAL7=WKCOSL(L)+BRCOSL(L)+RLCOSL(L)+BLCOSL(L)
              CNCOSL(L)=VAL1+VAL2+VAL3+VAL4+VAL5+VAL6+VAL7
              TCONST=TCONST+CNCOSL(L)
246           CONTINUE
*
      DO 247 I=1,4
              SPACE=COL+XSPA*I+1
              LENGTH=LEN('CONSTRUCTION COST')
              IFRONT=INT((XSPA-LENGTH)/2.0)+SPACE
              LINE3(IFRONT:IFRONT+LENGTH)='CONSTRUCTION COST'
247           CONTINUE
*
      LENGTH=LEN('LINK')
      IFRONT=INT((5-LENGTH)/2.0)+2
      LINE4(IFRONT:IFRONT+LENGTH)='LINK'
*
      LENGTH=LEN('CONSTRUCTION COST')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+1
      LINE4(IFRONT:IFRONT+LENGTH)='CONSTRUCTION COST'
*
      LENGTH=LEN('PER PAVEMENT AREA')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+1
      LINE4(IFRONT:IFRONT+LENGTH)='PER PAVEMENT AREA'
*
      LENGTH=LEN('PER STREET LENGTH')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*2+1
      LINE4(IFRONT:IFRONT+LENGTH)='PER STREET LENGTH'
*
      LENGTH=LEN('PER SALEABLE FRONTAGE')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*3+1
      LINE4(IFRONT:IFRONT+LENGTH)='PER SALEABLE FRONTAGE'
```

```
*
   LENGTH=LEN('PER DWELLING UNIT')
   IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*4+1
   LINE4(IFRONT:IFRONT+LENGTH)='PER DWELLING UNIT'
*
   XTOTAL=COL+XSPA*2
   LINE2(COL+XSPA:COL+XSPA)='+'
   DO 248 I=COL+XSPA+1,MCOL+COL
           IF ( I.NE.XTOTAL )  THEN
             LINE2(I:I)=' '
           ELSE
             XTOTAL=XTOTAL+XSPA
             LINE2(I:I)='!'
           ENDIF
248        CONTINUE
*
   LINE3(COL+XSPA:COL+XSPA)='!'
   LINE3(COL+XSPA*3:COL+XSPA*3)='!'
   LINE4(COL+XSPA:COL+XSPA)='!'
   LINE4(COL+XSPA*3:COL+XSPA*3)='!'
*
   WRITE (WUNIT1,591)
   DO 249 I=1,TIME
           WRITE (WUNIT1,595)
249        CONTINUE
*
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,400) LINE3
   WRITE (WUNIT1,400) LINE2
   WRITE (WUNIT1,400) LINE4
   WRITE (WUNIT1,400) LINE1
*
   YTOTAL=YSEC+1
   DO 250 I=1,NLINK
           IF ( I.EQ.YTOTAL )  THEN
             WRITE (WUNIT1,400) LINE1
             YTOTAL=YTOTAL+YSEC
           ENDIF
           IF ( ALINK(I).EQ.0.0 )  THEN
             RAT1=0.0
           ELSE
             RAT1=CNCOSL(I)/ALINK(I)
           ENDIF
           IF ( RDPINT(I).EQ.0.0 )  THEN
             RAT2=0.0
           ELSE
             RAT2=CNCOSL(I)/RDPINT(I)
           ENDIF
           IF ( FRONT(I).EQ.0.0 )  THEN
             RAT3=0.0
           ELSE
             RAT3=CNCOSL(I)/FRONT(I)
           ENDIF
           IF ( (MULTI(I)+UNITS(I)).EQ.0.0 )  THEN
             RAT4=0.0
           ELSE
             RAT4=CNCOSL(I)/(MULTI(I)+UNITS(I))
```

```
            ENDIF
            WRITE (WUNIT1,480) I,CNCOSL(I),RAT1,RAT2,RAT3,RAT4
250         CONTINUE
*
   WRITE (WUNIT1,400) LINE
   WRITE (WUNIT1,485) TCONST
   DO 255 I=1,TIME
            WRITE (WUNIT1,423)
255         CONTINUE
   WRITE (WUNIT1,400) LINE
*
*
*
*  summary
*
*
   WRITE (WUNIT1,900)
   DO 257 I=1,TIME
            WRITE (WUNIT1,905)
257         CONTINUE
   WRITE (WUNIT1,910)
   DO 258 I=1,TIME
            WRITE (WUNIT1,911)
258         CONTINUE
   WRITE (WUNIT1,915)
   DO 259 I=1,TIME
            WRITE (WUNIT1,920)
259         CONTINUE
*
   WRITE (WUNIT1,925) WMPIPE,WMCOST
   WRITE (WUNIT1,930) DSPIPE,DSCOST
   WRITE (WUNIT1,935) SSPIPE,SSCOST
   WRITE (WUNIT1,940) ACOL,CSCOST
   WRITE (WUNIT1,945) ALOC,LSCOST
   WRITE (WUNIT1,950) TPAREA,TCPAVE
   WRITE (WUNIT1,955) TWK,WKCOST
   WRITE (WUNIT1,960) TBR,BRCOST
   WRITE (WUNIT1,965) TRL,RLCOST
   WRITE (WUNIT1,970) TBL,BLCOST
   WRITE (WUNIT1,980) TSRCOS
   WRITE (WUNIT1,1063) APARK,PCOST
   WRITE (WUNIT1,1065) ABUFF,BCOST
   WRITE (WUNIT1,985) NBASIN,CBCOST
   WRITE (WUNIT1,990) TDSMAN,TDSCOS
   WRITE (WUNIT1,995) TSSMAN,TSSCOS
   WRITE (WUNIT1,1005) MNHOLE,MHCOST
   WRITE (WUNIT1,1010) NLIGHT,LTCOST
   WRITE (WUNIT1,1015) TSIGN,CSIGN
   WRITE (WUNIT1,1020) NTRAFF,TFCOST
   WRITE (WUNIT1,1030) INT(TSING)
   WRITE (WUNIT1,1035) GAS1
   WRITE (WUNIT1,1040) POW1
   WRITE (WUNIT1,1045) PH1
   WRITE (WUNIT1,1050) INT(TMULT)
   WRITE (WUNIT1,1035) GAS2
   WRITE (WUNIT1,1040) POW2
   WRITE (WUNIT1,1045) PH2
```

```
      WRITE (WUNIT1,1060) INT(TSING+TMULT),(TUSING+TUMULT)
      WRITE (WUNIT1,1070) TLCOST
      DO 260 I=1,TIME
              WRITE (WUNIT1,1080) TLCOST
260           CONTINUE
*
*
*
*  exit/cul-de-sac/intersections
      WRITE (WUNIT1,600)
      DO 261 I=1,TIME
              WRITE (WUNIT1,610)
261           CONTINUE
      WRITE (WUNIT1,615)
      DO 262 I=1,TIME
              WRITE (WUNIT1,616)
262           CONTINUE
*
      WRITE (WUNIT1,617)
      WRITE (WUNIT1,620)
      WRITE (WUNIT1,625) NCUL
      WRITE (WUNIT1,630) NDEAD
      WRITE (WUNIT1,635) N3WAY
      WRITE (WUNIT1,640) N4WAY
      WRITE (WUNIT1,645) NEXIT
      WRITE (WUNIT1,647)
      WRITE (WUNIT1,650)
      WRITE (WUNIT1,651) ASING
      WRITE (WUNIT1,652) AMULT
      WRITE (WUNIT1,653) ASCHOL
      WRITE (WUNIT1,654) APARK
      WRITE (WUNIT1,655) ABUFF
      WRITE (WUNIT1,656) ACOM
      WRITE (WUNIT1,657) APAVE
      WRITE (WUNIT1,658) AWALK
      WRITE (WUNIT1,659) AOTHER
      WRITE (WUNIT1,660)
      WRITE (WUNIT1,661) ASUB
      DO 265 I=1,TIME
              WRITE (WUNIT1,662)
265           CONTINUE
*
*  construction cost
      TCPTA=TLCOST/ASUB
      TCPGA=TLCOST/GROSS
      TCPST=TLCOST/TLEN
      TCPPV=TLCOST/APAVE
      TCPNA=TLCOST/(ASING+AMULT)
      TCPSA=TLCOST/ASING
      TCPSG=TLCOST/TSING
      TCPDW=TLCOST/(TSING+TMULT)
      IF ( TFRONT.GT.0.0 )  THEN
              TCPFG=TLCOST/TFRONT
      ELSE
              TCPFG=0.0
      ENDIF
      WRITE (WUNIT1,665)
```

```
      DO 270 I=1,TIME
              WRITE (WUNIT1,670)
270           CONTINUE
      WRITE (WUNIT1,675)
      DO 272 I=1,TIME
              WRITE (WUNIT1,676)
272           CONTINUE
      WRITE (WUNIT1,681) TCPTA
      WRITE (WUNIT1,682) TCPGA
      WRITE (WUNIT1,685) TCPNA
      WRITE (WUNIT1,687) TCPSA
      WRITE (WUNIT1,684) TCPPV
      WRITE (WUNIT1,683) TCPST
      WRITE (WUNIT1,686) TCPFG
      WRITE (WUNIT1,689) TCPDW
      WRITE (WUNIT1,688) TCPSG

*
*  density of the sub-division
      GSING=GROSS-AMULT
      GMULT=GROSS-ASING
      SIGPGA=TSING/GSING
      MULPGA=TMULT/GMULT
      TOTPGA=(TSING+TMULT)/GROSS
*
      SIGPNA=TSING/(ASING+AMULT)
      MULPNA=TMULT/(ASING+AMULT)
      TOTPNA=(TSING+TMULT)/(ASING+AMULT)
*
      WRITE (WUNIT1,690)
      DO 280 I=1,TIME
              WRITE (WUNIT1,700)
280           CONTINUE
      WRITE (WUNIT1,710)
      DO 282 I=1,TIME
              WRITE (WUNIT1,711)
282           CONTINUE
      WRITE (WUNIT1,720)
      DO 283 J=1,TIME
              WRITE(WUNIT1,721)
283           CONTINUE
      WRITE (WUNIT1,725)
      WRITE (WUNIT1,730) INT(TSING),GSING,SIGPGA
      WRITE (WUNIT1,735) INT(TMULT),GMULT,MULPGA
      WRITE (WUNIT1,740) INT(TSING+TMULT),GROSS,TOTPGA
      WRITE (WUNIT1,750)
      DO 284 J=1,TIME
              WRITE(WUNIT1,751)
284           CONTINUE
      WRITE (WUNIT1,760)
      WRITE (WUNIT1,730) INT(TSING),(ASING+AMULT),SIGPNA
      WRITE (WUNIT1,735) INT(TMULT),(ASING+AMULT),MULPNA
      WRITE (WUNIT1,740) INT(TSING+TMULT),(ASING+AMULT),TOTPNA

*
*
*
```

```
*  yield efficiency
   WRITE (WUNIT1,780)
   DO 290 I=1,TIME
          WRITE (WUNIT1,790)
290       CONTINUE
   WRITE (WUNIT1,795)
   DO 292 I=1,TIME
          WRITE (WUNIT1,796)
292       CONTINUE
   WRITE (WUNIT1,800) TFRONT
   WRITE (WUNIT1,805) TLEN
   WRITE (WUNIT1,810) GROSS
   WRITE (WUNIT1,815) YSTLEN
   DO 294 I=1,TIME
          WRITE (WUNIT1,825)
294       CONTINUE
   WRITE (WUNIT1,820) YGROSS


*
*
*
   TCOS=TCPFG+ASSUME
*  CALL GRAPH(REVMAX,REVMIN,TCOS)
   CALL GRAPH(3500.0,2500.0,2300.0)


400       FORMAT (1X,A132)
410       FORMAT (1X,'!TOTAL!',8X,F10.2,'m',10X,'!',29X
     &    ,'!',29X,'!',5X,'$',3X,F10.2,10X,'!')
420       FORMAT (1X,'!',1X,I3,1X,'!',3(6X,I4,9X,'!',2X,'$',F10.2,6X,'!'))
421       FORMAT (1X,'!TOTAL!',3(6X,I4,9X,'!',2X,'$',F10.2,6X,'!'))
423       FORMAT ('+',' TOTAL')
430       FORMAT (1X,'!',1X,I3,1X,'!',3(6X,I4,9X,'!',3X,'$',F10.2,5X,'!'))
431       FORMAT (1X,'!TOTAL!',3(6X,I4,9X,'!',3X,'$',F10.2,5X,'!'))
440       FORMAT (1X,'!',5X,I6,7X,'!',2(6X,I6,7X,'!'),59X,'!')
450       FORMAT (1X,'!',1X,I3,1X,'!',2(4X,I4,4X,'!',3(1X,'$'
     &    ,F9.2,1X,'!')),1X,'$',F9.2,1X,'!')
451       FORMAT (1X,'!TOTAL!',2(4X,I4,4X,'!',3(1X,'$'
     &    ,F9.2,1X,'!')),1X,'$',F9.2,1X,'!')
460       FORMAT (1X,'!',1X,I3,1X,'!',3(5X,F10.2,'m',7X,'!'),
     &    6X,F10.2,7X,'!',1X,'$',4X,F10.2,7X,'!')
461       FORMAT (1X,'!TOTAL!',3(5X,F10.2,'m',7X,'!'),
     &    6X,F10.2,7X,'!',1X,'$',4X,F10.2,7X,'!')
470       FORMAT (1X,'!',1X,I3,1X,'!',3(F10.2,'m',1X,'!',1X,F10.2,1X,'!$',
     &    F10.2,1X,'!'))
471       FORMAT (1X,'!TOTAL!',3(F10.2,'m',1X,'!',1X,F10.2,1X,'!$',
     &    F10.2,1X,'!'))
480       FORMAT (1X,'!',1X,I3,1X,'!',5X,'$',1X,F10.2,6X,'!',2X,F10.2
     &    ,' $/sq. m',3x,'!',2(4x,F10.2,' $/m',5X,'!'),2X,F10.2
     &    ,' $/units',3X,'!')
485       FORMAT (1X,'!TOTAL!',5X,'$',1X,F10.2,6X,'!',95X,'!')
500       FORMAT ('1',' S T O R M   S E W E R   S Y S T E M')
502       FORMAT ('+',' S T O R M   S E W E R   S Y S T E M')
504       FORMAT ('1',' D O M E S T I C   S E W E R   S Y S T E M')
506       FORMAT ('+',' D O M E S T I C   S E W E R   S Y S T E M')
508       FORMAT ('1',' W A T E R M A I N   S Y S T E M')
```

```
510         FORMAT ('+',' W A T E R M A I N   S Y S T E M')
520         FORMAT ('1',' M A N H O L E S   &   C A T C H - B A S I N S')
530         FORMAT ('+',' M A N H O L E S   &   C A T C H - B A S I N S')
540         FORMAT ('1',' S T R E E T   R E Q U I R E M E N T S')
545         FORMAT ('+',' S T R E E T   R E Q U I R E M E N T S')
550         FORMAT ('1',' P A V E M E N T   I N F O R M A T I O N')
555         FORMAT ('+',' P A V E M E N T   I N F O R M A T I O N')
560         FORMAT ('1',' F A M I L Y   U N I T S ')
570         FORMAT ('+',' F A M I L Y   U N I T S ')
580         FORMAT ('1',' S I D E W A L K   R E Q U I R E M E N T S')
590         FORMAT ('+',' S I D E W A L K   R E Q U I R E M E N T S')
591         FORMAT ('1',' C O N S T R U C T I O N   C O S T')
595         FORMAT ('+',' C O N S T R U C T I O N   C O S T')
600         FORMAT ('1',' S I T E   S P E C I F I C   I N F O R M A T I O N')
610         FORMAT ('+',' S I T E   S P E C I F I C   I N F O R M A T I O N')
615         FORMAT (1X,' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
616         FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
617         FORMAT (/////,1X,'Road Elements :')
620         FORMAT (//,6X,'DESCRIPTION',20X,'QUANTITY')
625         FORMAT (/,6X,'Cul-de-sacs',22X,I4)
630         FORMAT (/,6X,'Dead-ends',24X,I4)
635         FORMAT (/,6X,'Three-way intersections',10X,I4)
640         FORMAT (/,6X,'Four-way intersections',11x,I4)
645         FORMAT (/,6X,'Subdivision exits',16X,I4)
647         FORMAT (/////,1X,'Land Use Distribution :')
650         FORMAT (//,6X,'DESCRIPTION',24X,'AREA')
651         FORMAT (/,6X,'Net single family',12X,F10.2,' Ha')
652         FORMAT (/,6X,'Net multi-family',13X,F10.2,' Ha')
653         FORMAT (/,6X,'School',23X,F10.2,' Ha')
654         FORMAT (/,6X,'Park',25X,F10.2,' Ha')
655         FORMAT (/,6X,'Buffer',23X,F10.2,' Ha')
656         FORMAT (/,6X,'Commercial',19X,F10.2,' Ha')
657         FORMAT (/,6X,'Pavement',21X,F10.2,' Ha')
658         FORMAT (/,6X,'Sidewalk',21X,F10.2,' Ha')
659         FORMAT (/,6X,'Other',24X,F10.2,' Ha')
660         FORMAT (2X,'-----------------------',10X,'-------------')
661         FORMAT (/,3X,'Gross Subdivision Area',10X,F10.2,' Ha')
662         FORMAT ('+',2X,'Gross Subdivision Area')
665         FORMAT ('1',' S U B D I V I S I O N   C O N S T R U C T I O N
    &C O S T')
670         FORMAT ('+',' S U B D I V I S I O N   C O N S T R U C T I O N
    &C O S T')
675         FORMAT (1X, ' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    &~~~~~~~')
676         FORMAT ('+', ' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    &~~~~~~~')
681         FORMAT (/////,' Total construction cost per total subdivision
    &area = ',F10.2,' $/Ha',/)
682         FORMAT (/,29X,'gross area',13X,'= ',F10.2,' $/Ha')
683         FORMAT (/,29X,'street length',10X,'= ',F10.2,' $/m')
684         FORMAT (/,29X,'pavement area',10X,'= ',F10.2,' $/Ha',/)
685         FORMAT (/,29X,'net area',15X,'= ',F10.2,' $/Ha')
686         FORMAT (/,29X,'saleable frontage',6X,'= ',F10.2,' $/m',/)
687         FORMAT (/,29X,'single family area',5X,'= ',F10.2,' $/Ha')
688         FORMAT (/,29X,'single family',10X,'= ',F10.2,' $/units')
689         FORMAT (/,29X,'total dwelling unit',4X,'= ',F10.2,' $/units' )
690         FORMAT ('1',' S U B D I V I S I O N   D E N S I T Y')
```

```
700         FORMAT ('+',' S U B D I V I S I O N   D E N S I T Y')
710         FORMAT (1X,' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
711         FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
720         FORMAT (/////,1X,'Unit density based on GROSS area :')
721         FORMAT ('+',22X,'GROSS')
725         FORMAT (//,6X,'FAMILY UNIT',20X,'QUANTITY',20X,'GROSS AREA'
     &      ,20X,'DENSITY')
730         FORMAT (/,6X,'Single',21X,I6,' units',16X,F10.2,' Ha',
     &      13X,F10.2,' units/Ha')
735         FORMAT (/,6X,'Multi',22X,I6,' units',16X,F10.2,' Ha',
     &      13X,F10.2,' units/Ha')
740         FORMAT (/,6X,'Total',22X,I6,' units',16X,F10.2,' Ha',
     &      13X,F10.2,' units/Ha')
750         FORMAT (/////,1X,'Unit density based on NET area :')
751         FORMAT ('+',22X,'NET')
760         FORMAT (//,6X,'FAMILY UNIT',20X,'QUANTITY',21X,'NET AREA'
     &      ,21X,'DENSITY')
780         FORMAT ('1',' Y I E L D   E F F I C I E N C Y')
790         FORMAT ('+',' Y I E L D   E F F I C I E N C Y')
795         FORMAT (1X,' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
796         FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
800         FORMAT (///,6X,'Saleable frontage :',10X,F10.2,'m')
805         FORMAT (/,6X,'Street length :',14X,F10.2,'m')
810         FORMAT (/,6X,'Gross subdivision area :',5X,F10.2,' Ha')
815         FORMAT (///,6X,' YIELD - based on Total Street Length :',10X,
     &      F10.2,' m/m')
820         FORMAT (/,13X,'- based on Gross Subdivision Area :',7X,
     &      F10.2,' m/Ha')
825         FORMAT ('+',5X,' YIELD')
900         FORMAT ('1',' S U M M A R Y')
905         FORMAT ('+',' S U M M A R Y')
910         FORMAT (1X,' ~~~~~~~~~~~~~')
911         FORMAT ('+',' ~~~~~~~~~~~~~')
915         FORMAT (/////,6X,'DESCRIPTION',25X,'QUANTITY',40X,'COST')
920         FORMAT ('+',5X,'DESCRIPTION',25X,'QUANTITY',40X,'COST')
925         FORMAT (//,6X,'WATERMAIN',22X,F10.2,'m',34X,'$',1X,F10.2)
930         FORMAT (6X,'DOMESTIC SEWER',17X,F10.2,'m',34X,'$',1X,F10.2)
935         FORMAT (6X,'STORM SEWER',20X,F10.2,'m',34X,'$',1X,F10.2)
940         FORMAT (//,6X,'COLLECTOR STREET',15X,F10.2,'sq. m',30X,
     &      '$',1X,F10.2)
945         FORMAT (6X,'LOCAL STREET',19X,F10.2,'sq. m',30X,
     &      '$',1X,F10.2)
950         FORMAT (6X,'SUBTOTAL PAVEMENT',34X,F10.2,'sq. m',30X,
     &      '$',1X,F10.2)
955         FORMAT (//,6X,'SIDEWALK',23X,F10.2,'sq. m',30X,
     &      '$',1X,F10.2)
960         FORMAT (6X,'BARRIER CURB & GUTTER',10X,F10.2,'m',34X
     &      ,'$',1X,F10.2)
965         FORMAT (6X,'ROLLED CURB & GUTTER',11X,F10.2,'m',34X
     &      ,'$',1X,F10.2)
970         FORMAT (6X,'BLVD. CURB & GUTTER',12X,F10.2,'m',34X
     &      ,'$',1X,F10.2)
980         FORMAT (6X,'SUBTOTAL SIDEWALK REQUIREMENT',67X,
     &      '$',1X,F10.2)
985         FORMAT (//,6X,'CATCH-BASIN',23X,I4,2X,' units',30X,
     &      '$',1X,F10.2)
990         FORMAT (//,6X,'DOMESTIC SEWER MANHOLE',12X,I4,2X,' units',30X,
```

```
     &      '$',1X,F10.2)
995         FORMAT (6X,'STORM SEWER MANHOLE',15X,I4,2X,' units',30X,
     &      '$',1X,F10.2)
1005        FORMAT (6X,'SUBTOTAL MANHOLE',38X,I4,2X,' units',30X,
     &      '$',1X,F10.2)
1010        FORMAT (//,6X,'STREET LIGHT',22X,I4,2X,' units',30X,
     &      '$',1X,F10.2)
1015        FORMAT (6X,'STREET SIGN',23X,I4,2X,' units',30X,
     &      '$',1X,F10.2)
1020        FORMAT (6X,'TRAFFIC SIGN',22X,I4,2X,' units',30X,
     &      '$',1X,F10.2)
1030        FORMAT (//,6X,'SINGLE FAMILY',21X,I4,2X,' units')
1035        FORMAT (10X,'- GAS',67X,'$',1X,F10.2)
1040        FORMAT (10X,'- POWER',65X,'$',1X,F10.2)
1045        FORMAT (10X,'- TELEPHONE',61X,'$',1X,F10.2)
1050        FORMAT (6X,'MULTI-FAMILY',22X,I4,2X,' units')
1060        FORMAT (6X,'SUBTOTAL UTILITIES',36X,I4,2X,' units',29X,
     &      '$',1X,F10.2)
1063        FORMAT (//,6X,'PARK DEVELOPMENT',15X,F10.2,'Ha',33X,
     &      '$',1X,F10.2)
1065        FORMAT (6X,'BUFFER DEVELOPMENT',13X,F10.2,'Ha',33X,
     &      '$',1X,F10.2)
1070        FORMAT (//,11X,'T O T A L',62X,'$',1X,F10.2)
1080        FORMAT ('+',10X,'T O T A L',62X,'$',1X,F10.2)
*
*
   RETURN
   END
```

B.2.6 SUBROUTINE: ATTACH

```
      SUBROUTINE ATTACH


*****************************************************************
*                                                               *
*                        SUBROUTINE  ATTACH                     *
*                                                               *
*****************************************************************
*
*
*
*  XXXXXX -
*  ATTLIN - the links associate with the given node. ( NOTE: max. link can
*           attach to the given node is 4 )
*  ALINK  - the area of the road on a link.
*  AINTS  - area of intersection.
*  CORWD  - corresponding road width of the given link.
*  LROAD  - length of the road on a link (calculate).
*  DSLINK - domestic sewer elev. of a given link.
*  SSLINK - storm sewer elev. of a given link.
*  EXIT   - a flag indicates the exit node
*  L,N,J  - loop counter
*
*
*
      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &      ATTLIN,EXIT

      REAL
        &      LROAD

      COMMON
        &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
        &      ANGLE(NODNUM)
        &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
        &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
        &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
        &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
        &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
        &      /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
*
*
*
*  This is a supplementary program of the PAVEMENT routine,
*  it will set all the calculated variables to zero
*  before the PAVEMENT routine is called.
*  And it also find all the links that associate with the node.
*
*
*
*  clear the variables
```

```
      DO 100 L=1,NLINK
             LROAD(I)=0.0
             ALINK(L)=0.0
             DO 200 J=1,2
                    CORWD(L,J)=0.0
200                 CONTINUE
100          CONTINUE
*
*
*
      DO 300 N=1,NNODE
             DO 400 J=1,4
                    SSLINK(N,J)=9999.9
                    DSLINK(N,J)=9999.9
400                 CONTINUE
             EXIT(N)=0
             AINTS(N)=0.0
*            find the associate links to the given node
             CALL ADJ(N,-1,WD)
300          CONTINUE
*
*
*
      RETURN
      END
```

B.2.7 SUBROUTINE: CATBAS

SUBROUTINE CATBAS

```
*****************************************************************************
*                                                                           *
*                                                                           *
*                              CATCHBASIN                                   *
*                             REQUIREMENTS                                  *
*                                                                           *
*                                                                           *
*****************************************************************************
*
*
*
*  NTYPE  - node type.
*  ATTLIN - the links associate with the given node. ( NOTE: max. links can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  CDPTH  - depth of the basin lead from the ground level. ( value returned
*           from CATDEPTH routine )
*  CLEAD  - length of the catchbasin lead. ( value returned from
*           CATDEPTH routine )
*  AVEDEP - average depth of the storm sewer of the current link
*  WROAD  - width of the road of a given link.
*  CBDPTH - depth of the basin lead from the ground level.
*  CBLEAD - length of the catchbasin lead.
*  BASIN  - number of catch-basins required for current node.
*  FLINK  - flag indicates the drainage link.
*  CBCOS  - the cost of the catch-basins for current node.
*  NFLOW  - number of drainage flow for current node.
*  SSLINK - storm sewer elev. of a given link.
*  CBASIN - total number of catch-basins required in the sub-division.
*  CBCOST - total cost of the catch-basins
*  STRLEN - street length between two intersections or intersection/dead
*           end/cul-de-sac
*  SPACAT - spacing of catch-basin
*  BASHGT - the height between the lead and the bedding.
*  BASBED - the height of the bedding.
*  OFFSET - the distance from the center of the street.
*  NBASIN - number of catch-basins per street length ( city block )
*  STRCAT - number of catch-basins per street length ( city block )
*           ( record the values in the memory )
*  PATH   - the links between two intersections or intersection/dead
*           end/cul-de-sac
*  LBASIN - number of catch-basins required for current link ( include the
*           the catch-basins for associated node )
*  LCBCOS - the cost of the catch-basins for current link ( include the
*           the catch-basins for associated node )
*           routine )
*  NL     - number of links associsted with a given node
*  NBL    - number of links between two intersections or intersection/dead
*           end/cul-de-sac
*  BRANCH - number of possible paths
*  BULB   - a flag indicates the bulb curve(s) found.
```

```
*  FLAG    - a flag indicates the direction of the drainage flow.
*            when  FLAG = 0  means flow out
*                  FLAG = 1  means flow in
*  N1,N2   - loop counter
*  I,J,K   - loop counter
*  SS      - subscript
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  FLAG,CLINK,ATTLIN,BASIN,PATH,BRANCH,STRCAT

      INTEGER
     &  CBASIN,RUNIT1,RUNIT2,WUNIT1,WUNIT2,FLINK,BULB,ERRFLG

      REAL
     &  LROAD,CLEAD(2),LCBCOS

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
     &      /AREA19/TLCOST
     &      /AREA20/ERRFLG
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA25/SPAMAN,SPALIT,SPACAT
     &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
     &      /AREA30/CBLEAD(4,2),CBDPTH(4),FLINK(NODNUM,4),
     &      BASIN(NODNUM),CBCOS(NODNUM),LBASIN(LINNUM),LCBCOS(LINNUM)
     &      ,STRCAT(NODNUM,NODNUM,4)
     &      /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
     &      /AREA33/BASHGT,BASBED,OFFSET
     &      /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
     &      MHCOST,CBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
     &      WKCOST,BRCOST,RLCOST,BLCOST
*
*
*
*  TEMPORARY HEIGHT OF THE BASIN
      BASHGT=.45
      BASBED=.20
      OFFSET=3.6576
      SPACAT=300*12*2.54/100
*
*
*
```

```
*  set the total to zero
   CBASIN=0
   CBCOST=0.0
*
   DO 90 I=1,NNODE
*  set the sub-total ( per node ) to zero
            CBCOS(I)=0.0
            BASIN(I)=0
            DO 95 N=1,NNODE
              DO 100 J=1,4
*  set the flow link flag to zero
                FLINK(I,J)=0
*  set the number of catch-basins per street length ( city block ) to zero
                STRCAT(I,N,J)=0
100                   CONTINUE
95          CONTINUE
90 CONTINUE
*
   DO 110 L=1,NLINK
*  set the sub-total ( per link ) to zero
            LBASIN(L)=0
            LCBCOS(L)=0.0
*  set the average depth to zero
            AVEDEP(L)=0.0
110         CONTINUE
*
*
*
*
   DO 120 I=1,NNODE
*  determine the number of catch-basin per every intersection
*  and the cost of the catch-basins
            IF ( NTYPE(I).EQ.3 .OR. NTYPE(I).EQ.4 )  THEN
*
*
*  determine the number of catch-basin per every intersection
*  set the flow counter to zero
              NFLOW=0
*
*  set the link counter to zero
              NL=0
*
*  find out which associated links drain into the current node
              DO 130 J=1,4
                CLINK=ATTLIN(I,J)
                IF ( CLINK.EQ.0 )  GOTO 130
*  COUNT THE NUMBER OF LINK ATTACHED TO NODE
                NL=NL+1
*  ELIMINATE THE EASEMENT
                IF ( WROAD(CLINK).EQ.0.0 )  GOTO 130
                CALL DRAIN(I,CLINK,FLAG)
                IF ( FLAG.NE.-1 )  THEN
                  NFLOW=NFLOW+FLAG
                  IF ( FLAG.EQ.1 )  THEN
                    FLINK(I,J)=1
                  ENDIF
                ELSE
```

```
                  FLINK(I,J)=-1
                  IF ( ERRFLG.EQ.0 ) THEN
                    WRITE(5,350)'CATBAS-W-No drainage ( flat surface
     & ) for link #',CLINK
                  ENDIF
                  IF ( ERRFLG.EQ.1 ) THEN
                    WRITE(WUNIT2,350)'CATBAS-W-No drainage ( flat
     &surface ) for link #',CLINK
                  ENDIF
                  IF ( ERRFLG.EQ.2 ) THEN
                    WRITE(5,350)'CATBAS-W-No drainage ( flat surface
     & ) for link #',CLINK
                    WRITE(WUNIT2,350)'CATBAS-W-No drainage ( flat
     &surface ) for link #',CLINK
                  ENDIF
                ENDIF
130                 CONTINUE
*
*  check the drainage condition
*  and add enough catch-basins to the node
              IF ( NFLOW.NE.0 )  THEN
                IF ( NFLOW.NE.NTYPE(I) )  THEN
                  BASIN(I)=NFLOW+1
                ELSE
*  does the node associate with an easement?
                  IF ( NL.EQ.NTYPE(I) )  THEN
                    BASIN(I)=-1
                    IF ( ERRFLG.EQ.0 ) THEN
                      WRITE(5,355)'CATBAS-W-No drainage exit at node #',I
                    ENDIF
                    IF ( ERRFLG.EQ.1 ) THEN
                      WRITE(WUNIT2,355)'CATBAS-W-No drainage exit at
     &node #',I
                    ENDIF
                    IF ( ERRFLG.EQ.2 ) THEN
                      WRITE(5,355)'CATBAS-W-No drainage exit at node #',I
                      WRITE(WUNIT2,355)'CATBAS-W-No drainage exit at
     &node #',I
                    ENDIF
                  ELSE
*  there is an easement
                    BASIN(I)=NFLOW
                  ENDIF
                ENDIF
              ELSE
                BASIN(I)=0
              ENDIF
*
*
*  calculate the cost of the catch-basins per node
*
*  determine all the possible length of the catch-basin lead
              DO 140 J=1,4
                CLINK=ATTLIN(I,J)
                IF ( CLINK.EQ.0 .OR. WROAD(CLINK).EQ.0.0 )  THEN
                  CBLEAD(J,1)=9999999.99
                  CBLEAD(J,2)=9999999.99
```

```
                CBDPTH(J)=0.0
              ELSE
                CALL CATDEPTH(I,CLINK,0,0.0,CLEAD,CDPTH)
                CBLEAD(J,1)=CLEAD(1)
                CBLEAD(J,2)=CLEAD(2)
                CBDPTH(J)=CDPTH
              ENDIF
140                 CONTINUE
*
*
*  check the number of catch-basins required
*  and determine the price of the catch-basins
*                 no catch-basins required
            IF ( BASIN(I).LE.0 )  THEN
              CBCOS(I)=0.0
            ELSE
*
              CALL CATCALC(I,NFLOW)
            ENDIF


*****************************************************************
          WRITE (088,990) I
          WRITE (088,991) (FLINK(I,J),J=1,4)
          WRITE (088,994) (ATTLIN(I,J),J=1,4)
          WRITE (088,995) (SSLINK(I,J),J=1,4)
          DO 999 J=1,4
          IF ( CBLEAD(J,1).GT.99999.0 ) THEN
          WRITE (088,993) ATTLIN(I,J)
          ELSE
          WRITE (088,992) ATTLIN(I,J),(CBLEAD(J,NQ),NQ=1,2)
          ENDIF
999               CONTINUE


990       FORMAT (/,1X,'AT NODE :',3X,I3)
991       FORMAT (10X,'LINKS HAVE FLOW IN :',4(3X,I3))
992       FORMAT (10X,'LINK :',3X,I3,3X,'LEAD LENGTH :',2(3X,F9.3))
993       FORMAT (10X,'LINK :',3X,I3,3X,'LEAD LENGTH :',2(3X,' --------'))
994       FORMAT (10X,'ATTACHED LINKS :',4(3X,I3))
901       FORMAT (5X,'FIND THE CORRECT PERP. LINK',//)
995       FORMAT (10X,'S. S. LINK ELEV. :',4(3X,F9.3))

*****************************************************************
          ENDIF
*
*
*
*  add one catch-basin to the cul-de-sac/dead end
*  and find the cost of the catch-basin
*
          IF ( NTYPE(I).EQ.1 )  THEN
            CALL CATNODE(I)
          ENDIF
120       CONTINUE
*
```

```
*
*
*  determine the number of catch-basins per link
*  and the cost of the catch-basins
*
   DO 150 N1=1,NNODE
           DO 160 N2=1,N1
*
*
*  set the number of catch-basins per street length ( or city block )
*  to zero
             NBASIN=0
             IF ( N1.EQ.N2 )  THEN
               BRANCH=1
             ELSE
               BRANCH=4
             ENDIF
             DO 165 J=1,4
               IF ( STRLEN(N1,N2,J).NE.0.0 )  THEN
                 IF ( BRANCH.GT.0 )  THEN
*
                   BRANCH=BRANCH-1
*  set the number of links between two intersections or intersection/dead
*  end/cul-de-sac to zero
                   NBL=0
*  determine the number of links between two intersections or
*  intersection/dead end/cul-de-sac
170                        NBL=NBL+1
                   CLINK=PATH(N1,J,NBL)
                   IF ( CLINK.EQ.0 )  THEN
                     NBL=NBL-1
                   ELSE
                     GOTO 170
                   ENDIF
*
*  corner check
                   CALL CORCHK(N1,J,NBL,BULB)
*
                   IF ( BULB.EQ.0 )  THEN
                           CALL CATLIN(N1,N2,J,1,NBL,0,0.0)
                   ELSE
                           CALL CORNER(N1,N2,J,NBL)
                   ENDIF
                 ENDIF
               ENDIF
165                  CONTINUE
160                CONTINUE
150        CONTINUE
*
*
*
*  add up the total
   DO 190 I=1,NLINK
           CBCOST=LCBCOS(I)+CBCOST
           CBASIN=LBASIN(I)+CBASIN
190        CONTINUE
*
```

```
      CALL ASPLIT(11,CBCOST)
      TLCOST=TLCOST+CBCOST
*     WRITE(WUNIT1,320)CBASIN,CBCOST
320         FORMAT(10X,'CATCH-BASIN (#)',T40,I6,T54,F7.0,/)
350         FORMAT(1X,A49,I4)
355         FORMAT(1X,A36,I4)
*
*
*
*************************************************************************


*     OUTPUT THE RESULT (SPECIAL)
      DO 911 N1=1,NNODE
      DO 902 N2=1,N1
            DO 903 J=1,4
            IF ( STRLEN(N1,N2,J).NE.0.0 )  THEN
            WRITE (088,950) N1,N2,J,STRLEN(N1,N2,J),STRCAT(N1,N2,J)
            ENDIF
903                 CONTINUE
902         CONTINUE
911         CONTINUE
      DO 904 I=1,NNODE
      WRITE (088,900)I,BASIN(I),CBCOS(I)
904         CONTINUE
      DO 905 I=1,NLINK
      WRITE (088,931)I,LBASIN(I),LCBCOS(I)
905         CONTINUE

            WRITE (088,920)CBASIN
            WRITE (088,910)CBCOST

920         FORMAT ('1',' BASIN =',3X,I3)
910         FORMAT (//,' BASIN COST =',3X,F10.3)
931         FORMAT (1X,'LINK',2(3X,I3),F9.3)
900         FORMAT (1X,'NODE',2(3X,I3),F9.3)
950         FORMAT (' ','FROM NODE ',3X,I3,3X,' TO NODE ',3X,I3,3X,' BRANCH '
     &,3X,I3,3X,' LENGTH ',3X,F9.3,3X,'BASIN ',3X,I3)
      RETURN
      END
```

B.2.8 SUBROUTINE: CLEAR

```
      SUBROUTINE CLEAR

      INTEGER ESC,BLA,J,Q,SH

      ESC=27
      BLA=91
      J=74
      Q=63
      SH=104

      WRITE(5,100) ESC,BLA,2,J
      WRITE(5,110) ESC,BLA,Q,6,SH

      PRINT*,'RESIDENTIAL AREA COST ANALYSIS MODEL (R  A  C  A  M)'
      PRINT*,'-----------------------------------------------------'
      PRINT*

110         FORMAT ('+',3A,I1,A)
100         FORMAT ('+',2A,I1,A)

      RETURN
      END
```

B.2.9 SUBROUTINE: CREATE

```
      SUBROUTINE CREATE
*     ***********************************************************
*     *      This subroutine organizes 'to' and 'from' nodes,     *
*     *     determines which links are involved in each of the    *
*     *      infra structure systems, and associates links to     *
*     *              create the TBTRY-, -W, -D, -S data.          *
*     ***********************************************************
*     * NLINK =number of links.
*     * ELEV  =ground elevations of nodes.
*     * NODE1,NODE2=nodes describing links.
*     * TBTRY =returned values from GROUP1 subroutine.
*     * X,X1-X4=miscellaneous use.
*     * CHONUM=number of choices made for a given subsystem
*     * CHOLNK=link identifiers of links a choice was made over.
*     *       -(index,1-3).
*     *       -1 is the name of the link not chosen.
*     *       -2 is the name of the link chosen.
*     *       -3 is the name of the node at which the choice was made.

*     * NWMLNK=number of watermain links.
*     * WLINKS=links involved in watermain system.
*     * NWMGRP=number of tributary groups in watermain system.
*     * TBTRYW=watermain system 'to'/'from' link associations.

*     * NDALNK=number of domestic sewer links.
*     * DLINKS=links involved in domestic sewer system.
*     * NDSGRP=number of tributary groups in domestic sewer system.
*     * TBTRYD=domestic sewer system 'to'/'from' link associations.

*     * NSSLNK=number of storm sewer links.
*     * SLINKS=links involved in storm sewer system.
*     * NSSGRP=number of tributary groups in storm sewer system.
*     * TBTRYS=storm sewer system 'to'/'from' link associations.

      PARAMETER
       &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
       &  WLINKS,DLINKS,SLINKS,CHONUM,CHOLNK(25,3),TBTRYW,TBTRYD,
       &  TBTRYS,ERRFLG,TRIFLG,RUNIT1,RUNIT2,WUNIT1,WUNIT2,X

      COMMON
       &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
       &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
       &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
       &      /AREA6/NWMLNK,WLINKS(LINNUM),NWMGRP,TBTRYW(LINNUM,4)
       &  /AREA7/NDSLNK,DLINKS(LINNUM),NDSGRP,TBTRYD(LINNUM,4)
       &      /AREA8/NSSLNK,SLINKS(LINNUM),NSSGRP,TBTRYS(LINNUM,4)
       &      /AREA20/ERRFLG,ICDFLG,IDAFLG,TRIFLG
       &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2

*     ***********
*     * Organize nodes:  higher nodes in NODE1 and lower nodes in NODE2.
*     ***********
```

```
      DO 10, L=1, NLINK
      IF(ELEV(NODE2(L)).GT.ELEV(NODE1(L)))THEN
      X=NODE1(L)
      NODE1(L)=NODE2(L)
      NODE2(L)=X
      ENDIF
   10 CONTINUE

*     ***********
*     * Create TBTRYW data.
*     ***********

      CALL GROUP1(NWMLNK,WLINKS,CHONUM,CHOLNK,NWMGRP,TBTRYW)

*     ***********
*     * Create TBTRYD data.
*     ***********

      CALL GROUP2(NDSLNK,DLINKS,CHONUM,CHOLNK,NDSGRP,TBTRYD)

*     * Inform the user of any choices in links that were made.
      IF(CHONUM.NE.0)THEN
       DO 20, L=1, CHONUM
       IF(ERRFLG.EQ.0) WRITE(5,200) CHOLNK(L,1),CHOLNK(L,2),
        &    CHOLNK(L,3)
       IF(ERRFLG.EQ.1) WRITE(WUNIT2,200) CHOLNK(L,1),CHOLNK(L,2),
        &    CHOLNK(L,3)
       IF(ERRFLG.EQ.2) THEN
              WRITE(5,200) CHOLNK(L,1),CHOLNK(L,2),CHOLNK(L,3)
              WRITE(WUNIT2,200) CHOLNK(L,1),CHOLNK(L,2),CHOLNK(L,3)
       ENDIF
   20  CONTINUE
      ENDIF

*     * Clear CHONUM and CHOLNK
      DO 30, L=1, 25
      CHOLNK(L,1)=0
      CHOLNK(L,2)=0
      CHOLNK(L,3)=0
   30 CONTINUE
      CHONUM=0

*     ***********
*     * Create TBTRYS data.
*     ***********

      CALL GROUP2(NSSLNK,SLINKS,CHONUM,CHOLNK,NSSGRP,TBTRYS)

*     * Inform the user of any choices in links that were made.
      IF(CHONUM.NE.0)THEN
       DO 40, L=1, CHONUM
       IF(ERRFLG.EQ.0) WRITE(5,201) CHOLNK(L,1),CHOLNK(L,2),
        &    CHOLNK(L,3)
       IF(ERRFLG.EQ.1) WRITE(WUNIT2,201) CHOLNK(L,1),CHOLNK(L,2),
        &    CHOLNK(L,3)
       IF(ERRFLG.EQ.2) THEN
              WRITE(5,201) CHOLNK(L,1),CHOLNK(L,2),CHOLNK(L,3)
```

```
            WRITE(WUNIT2,201) CHOLNK(L,1),CHOLNK(L,2),CHOLNK(L,3)
      ENDIF
40    CONTINUE
     ENDIF

*     * Do the following output only if the flag indicates that it
*     * should be done.
     IF(TRIFLG.EQ.0)THEN

*     ***********
*     * Output Watermain tributary data.
*     ***********
      WRITE(WUNIT1,202)
      X=1
      DO WHILE((TBTRYW(X,1).NE.0).AND.(X.LE.150))
       WRITE(WUNIT1,203) ((TBTRYW(X,L)),L=1,4)
       X=X+1
      ENDDO

*     ***********
*     * Output Domestic Sewer tributary data.
*     ***********
      WRITE(WUNIT1,204)
      X=1
      DO WHILE((TBTRYD(X,1).NE.0).AND.(X.LE.150))
       WRITE(WUNIT1,203) ((TBTRYD(X,L)),L=1,4)
       X=X+1
      ENDDO

*     ***********
*     * Output Storm Sewer tributary data.
*     ***********
      WRITE(WUNIT1,206)
      X=1
      DO WHILE((TBTRYS(X,1).NE.0).AND.(X.LE.150))
       WRITE(WUNIT1,203) ((TBTRYS(X,L)),L=1,4)
       X=X+1
      ENDDO

     ENDIF

200         FORMAT(1X,'CREATE-I-A path choice was made in domestic sewer',
     &  ' system:',/1X,T5,'- Link not chosen - #',I3,
     &  /1X,T5,'- Link chosen     - #',I3,
     &  /1X,T5,'- Location (node) - #',I3,/)

201         FORMAT(1X,'CREATE-I-A path choice was made in storm sewer',
     &  ' system:',/1X,T5,'- Link not chosen - #',I3,
     &  /1X,T5,'- Link chosen     - #',I3,
     &  /1X,T5,'- Location (node) - #',I3,/)

202         FORMAT('1','---   WATERMAIN TRIBUTARY DATA  ---'/,'    Feed Links
     &    Receptor link')
203         FORMAT(2X,3I4,5X,'-',3X,I4)


204         FORMAT('1','---   DOMESTIC SEWER TRIBUTARY DATA  ---'/,'      Fee
```

```
     & Links        Receptor link')
205        FORMAT(3X,3I4,4X,'-',3X,I4)


206        FORMAT('1','-     STORM SEWER TRIBUTARY DATA    -'/,'        Feed
     &Links       Receptor link')
207        FORMAT(5X,3I4,4X,'-',3X,I4)

   END
```

B.2.10 SUBROUTINE: DMSTC

```
      SUBROUTINE DMSTC(DSCOST)
*****************************************************************
*                                                               *
*                        SUBROUTINE DMSTC                       *
*                                                               *
*   This subroutine designs the DOMESTIC SEWER system. For each *
*   link, the subroutine: (1) estimates the generated domestic  *
*   sewer flow, (2) calculates the slope and the pipe size needed *
*   to carry the estimated sewer flow, (3) selects a standard sized *
*   domestic sewer pipe based on the calculated diameter, (4)   *
*   verifies the maximum and minimum allowable water velocities, *
*   (5) adjusts the pipe slope and/or the pipe size to meet the *
*   tolerances set by the maximum and minimum allowable water   *
*   velocities, and (6) calculates the average depth of ground  *
*   cover over the domestic sewer.                              *
*                                                               *
*****************************************************************
*
*  DSCOVR  - MINIMUM DEPTH OF COVER FOR DOMESTIC SEWER (M)
*  DSDPTH  - DEPTH OF DOMESTIC SEWER (M)
*  DSCOST  - TOTAL COST OF DOMESTIC SEWER ($)
*  QPCDS   - DOMESTIC SEWER FLOW PER CAPITA
*            (455 LITRES PER CAPITA CONVERTED TO CMS PER CAPITA
*            AS OBTAINED FROM HAMILTON/WENTWORTH)
*  SLOPE   - SLOPE BETWEEN NODE1 AND NODE2
*  DSLOPE  - SLOPE OF THE DOMESTIC SEWER PIPE BETWEEN THE ORIGIN
*            NODE AND THE DESTINATION NODE
*  DSELEV  - ELEVATION OF DOMESTIC SEWER PIPE (M)
*  TDAREA  - TOTAL DRAINAGE AREA OF EACH DESTINATION LINK (ACRES)
*  A1      - TOTAL DRAINAGE AREA OF EACH LINK (ACRES)
*  X1      - TOTAL DRAINAGE AREA OF EACH LINK (HECTARES)
*  QDS     - TOTAL GENERATED SEWER FLOW (CFS)
*  D       - CALCULATED DIAMETER OF PIPE (M)
*  D2      - REQUIRED DIAMETER OF DOMESTIC SEWER PIPE (M)
*  DIADS   - DIAMETER OF DOMESTIC SEWER PIPE (mm)
*  INDX2   - DIAMETER INDEX FOR DOMESTIC SEWER (1 - 5)
*  DINDX2  - DEPTH INDEX FOR DOMESTIC SEWER (1 - 9)
*  CDS     - THE PER UNIT COST OF DOMESTIC SEWER FOR EACH LINK ($)
*
      PARAMETER
     &      NODNUM=100,LINNUM=150,INTNUM=10,
     &      QPCDS=455.0/(24.0*60.0*60.0*1000.0)
      INTEGER DINDX2,TBTRYD,WUNIT1,WUNIT2,RUNIT1,RUNIT2,CONST,ERRFLG
      REAL LLINK,MULTI
      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA7/NDSLNK,DLINKS(LINNUM),NDSGRP,TBTRYD(LINNUM,4)
     &      /AREA9/IQ,L,P(LINNUM),PDS(LINNUM),ASS(LINNUM)
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNU
     &      /AREA20/ID1FLG,ID2FLG,OD1FLG,OD2FLG,WATFLG,DMSFLG,STMFLG,MANFLG,
```

```
     &      SDWFLG,STRFLG,STLFLG,STSFLG,UTIFLG,PARFLG,OPRFLG,ECOFLG,ERRFLG
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA22/PEOPLE,WVMIN,FRICTN,USE,FLOAD
     &      /AREA23/DSCOVR,DSVMIN,DSVMAX


   DSCOST=0.0
   WRITE(WUNIT1,215)DSVMIN,DSVMAX,DSCOVR
215        FORMAT('1DOMESTIC SEWER',/X,14('='),//,
     &      3X,'MINIMUM ACCEPTABLE PIPE VELOCITY (M/S) = ',F7.3,/
     &      3X,'MAXIMUM ACCEPTABLE PIPE VELOCITY (M/S) = ',F7.3,//,
     &      3X,'MINIMUM DEPTH OF GROUND COVER OVER PIPE (M) = ',F7.3,
     &      //T83'DEPTH OF EXCAVATION',/,T9,'TRIBUTARY',T36,
     &      'GROUND',T46,'CALCULATED',T72,'PIPE',T83,'-------------------'
     &      ,T107'AVERAGE'/' LINK'T9'POPULATION   FLOW RATE     SLOPE'
     &      T47'DIAMETER'T59'VELOCITY'T72'SLOPE'T84'NODE1'T96'NODE2'T108
     &      'DEPTH'T119,'LINK'/,' (#)',T12,'(#)',T24,'(CMS)'T36'(M/M)',
     &      T49'(M)'T60,'(M/S)',T72,'(M/M)',T85,'(M)',T97,'(M)',T109,'(M)',
     &      T119,'(#)',/)
   DO 229 I=1,NDSGRP
          TPOP=0.0
          DO 228 J=1,3
                 CONST=TBTRYD(I,J)
                 IF(CONST.LE.0) THEN
                        GOTO 228
                 ENDIF
                 TOLINK=0.0
                 DO 216 K=1,NDSGRP
                        IF(CONST.EQ.TBTRYD(K,4)) THEN
                               TOLINK=1
                        ENDIF
216                     CONTINUE
                 P1=(UNITS(CONST)+MULTI(CONST))*PEOPLE+PDS(CONST)
                 TPOP=TPOP+P1
*
*              CALCULATE THE GENERATED DOMESTIC SEWER FLOW
*              USING THE FOLLOWING EQUATIONS OBTAINED FROM
*              City of Hamilton/Wentworth, Dept. of Engineering
*              Subdivision Consulting Engineer's Guide, 1982
*
                 FM=5.0/((P1/1000)**0.2)
                 IF(FM.LT.2.0) THEN
                        FM=2.0
                 ELSEIF (FM.GT.5.0) THEN
                        FM=5.0
                 ENDIF
                 QDS=P1*QPCDS*FM
                 E1=ELEV(NODE1(CONST))
                 E2=ELEV(NODE2(CONST))
                 GSLOPE=(E1-E2)/LLINK(CONST)
                 IF(GSLOPE.LE.0) THEN
                        SLOPE=0.000169
                 ELSE
                        SLOPE=GSLOPE
                 ENDIF
                 DSLOPE=SLOPE
                 D=(3.208*QDS*FRICTN/SQRT(DSLOPE))**0.375
```

```
                    DD=D
*
*               CHOOSE A PIPE DIAMETER THE SAME SIZE OR LARGER
*               THAN THE CALCULATED DIAMETER REQUIRED
*
217                         IF (DD.LT.0.200) THEN
                            D2=0.200
                            INDX2=1
                    ELSEIF(DD.LT.0.250) THEN
                            D2=0.250
                            INDX2=2
                    ELSEIF (DD.LT.0.300) THEN
                            D2=0.300
                            INDX2=3
                    ELSE
                            D2=0.375
                            INDX2=4
                    ENDIF
                    DIADS(CONST)=D2
                    CALL DIAMTR(DIADS(CONST),QDS,DSLOPE,FRICTN,DSVMIN
     &                      ,DSVMAX,VEL)
                    IF((VEL.GT.DSVMAX).AND.(D2.LT.0.375)) THEN
                            DD=D2
                            DSLOPE=SLOPE
                            GOTO 217
                    ENDIF
                    DSDPTH=DSCOVR+DIADS(CONST)+0.100
                    DSE1=ELEV(NODE1(CONST))-DSDPTH
                    IF(TOLINK.GT.0.5) THEN
                            GOTO 219
                    ENDIF
218                         DSE2=DSE1-LLINK(CONST)*DSLOPE
                    IF(DSELEV(NODE2(CONST)).GT.DSE2) THEN
                            DSELEV(NODE2(CONST))=DSE2
                    ENDIF
                    CALL DEPTH (ELEV(NODE1(CONST)),
     &                      ELEV(NODE2(CONST)),
     &                      DSE1,DSE2,
     &                      DPTH1,DPTH2,ADPTH,DINDX2)
                    IF((DPTH2*1.001).LT.DSDPTH) THEN
                            DSE1=DSE1-DSDPTH+DPTH2
                            GOTO 218
                    ENDIF
                    GOTO 220
219                         IF(DSELEV(NODE1(CONST)).GT.DSE1) THEN
                            DSELEV(NODE1(CONST))=DSE1
                    ENDIF
                    DSE2=DSELEV(NODE1(CONST))-LLINK(CONST)*DSLOPE
                    IF(DSELEV(NODE2(CONST)).GT.DSE2) THEN
                            DSELEV(NODE2(CONST))=DSE2
                    ENDIF
                    CALL DEPTH (ELEV(NODE1(CONST)),
     &                      ELEV(NODE2(CONST)),
     &                  DSELEV(NODE1(CONST)),DSE2,
     &                      DPTH1,DPTH2,ADPTH,DINDX2)
                    IF((DPTH2*1.001).LT.DSDPTH) THEN
                            DSELEV(NODE1(CONST))=DSELEV(NODE1(CONST))-
```

```
     &                                DSDPTH+DPTH2
                              GOTO 219
                      ENDIF
220                           CDS(CONST)=COST12(DINDX2,INDX2)
                      DSCOST=DSCOST+CDS(CONST)*LLINK(CONST)
                      DIADS(CONST)=DIADS(CONST)*1000.0
                      WRITE(WUNIT1,227)CONST,P1,QDS,GSLOPE,D,VEL,DSLOPE,DPTH1,
     &                        DPTH2,ADPTH,CONST
*
*
*  save the depth of excavation

            CALL DSSDPTH(CONST,DPTH1,DPTH2)

227                           FORMAT(I4,X,F10.0,2X,8(2X,F10.5)4X,I4)
228                   CONTINUE
            PDS(TBTRYD(I,4))=PDS(TBTRYD(I,4))+TPOP
229         CONTINUE
   CALL ASPLIT(2,DSCOST)
   RETURN
   END
```

B.2.11 SUBROUTINE: ECONOM

```
      SUBROUTINE ECONOM

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER WUNIT1,TYPE,TIME

      REAL MNCOST,INTRST

      COMMON
        &      /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
        &      /AREA19/TLCOST,TMCOST,TOCOST
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &      ,MNCOST(25,3),MCAPAF(25,10,3)
        &      /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &      ,OPCOST(25,3),OCAPAF(25,10,3)
*
*
      TIME=3
*
*  add up the annual operation and maintenance cost
      TOCOST=0.0
      TMCOST=0.0
      DO 100 TYPE=1,25
              TOCOST=OPCOST(TYPE,2)+TOCOST
              TMCOST=MNCOST(TYPE,2)+TMCOST

100           CONTINUE
      TACOST=TMCOST+TOCOST
*
      WRITE (WUNIT1,400)
      DO 110 I=1,TIME
              WRITE(WUNIT1,410)
110           CONTINUE
      WRITE (WUNIT1,420)
      DO 120 I=1,TIME
              WRITE(WUNIT1,430)
120           CONTINUE
      WRITE (WUNIT1,440) TLCOST
      WRITE (WUNIT1,455) TMCOST
      WRITE (WUNIT1,450) TOCOST
*
      WRITE (WUNIT1,460)
      DO 130 I=1,TIME
              WRITE (WUNIT1,470)
130           CONTINUE
*
*  calculate the annual and present worth
      DO 140 N=1,NINT
              PW=UNFORM1(INTRST(N),LIFE(N),TACOST)+TLCOST
              AC=UNFORM(INTRST(N),LIFE(N),TLCOST)+TACOST
              WRITE(WUNIT1,480)INTRST(N),LIFE(N),PW,AC
140           CONTINUE
*
```

```
*  calculate present worth and annual costs at an infinite life
*  at the last given interest rate
*
   RI=INTRST(NINT)/100
   PW=TACOST/RI+TLCOST
   AC=TLCOST*RI+TACOST
   WRITE(WUNIT1,490)INTRST(NINT),PW,AC
990        FORMAT(T9,F6.2,T20,'INFINITE',T33,F9.0,T49,F9.0)
*
*
400        FORMAT ('1',' D E T A I L E D   A N A L Y S I S')
410        FORMAT ('+',' D E T A I L E D   A N A L Y S I S')
420        FORMAT (1X,' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
430        FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
440        FORMAT (//,' CONSTRUCTION COST',5X,':',5X,'$',2X,F10.2)
450        FORMAT (/,' ANNUAL OPERATION COST :',5X,'$',2X,F10.2)
455        FORMAT (/,' ANNUAL MAINTENANCE COST :',3X,'$',2X,F10.2)
460        FORMAT (/////,12X,'INTEREST',14X,'LIFE',12X,'PRESENT WORTH'
     &     ,9X,'ANNUAL COST')
470        FORMAT ('+',11X,'INTEREST',14X,'LIFE',12X,'PRESENT WORTH'
     &     ,9X,'ANNUAL COST')
480        FORMAT (12X,F6.2,1X,'%',12X,I5,1X,'yr',8X,'$',2X,F12.2,
     &     6X,'$',2X,F11.2)
490        FORMAT (12X,F6.2,1X,'%',12X,'INFINITE',8X,'$',2X,F12.2,
     &     6X,'$',2X,F11.2)
*
*
*
   RETURN
   END
```

B.2.12 SUBROUTINE: ERRCHK

```
      SUBROUTINE ERRCHK
*     ***********************************************************
*     * This subroutine checks the validity of much of the input. *
*     ***********************************************************
*     * CHEKED=keeps track of which links have been cleared.
*     * CHEKIN=keeps track of which links have been passed during
*     *        the test of the current link.
*     * REFLNK=the link the test is currently referencing - this starts
*     *        at the link being tested, and moves along the path of
*     *        the test.
*     * BRANUM=the number of branches (forks in the path) passed.
*     * BRALNK=holds the link numbers before each branch.
*     * NUMPTH=number of possible paths test can follow.
*     * PATHS =a list of the link number of each path.
*     * FLAG1 =signifies an end to the test for a given link.
*     * FLAG2 =signifies a return to a former branch location
*     *        and recalculation of possible paths.
*     * X,X1,X2=miscellaneous variables.

      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER CHEKED(LINNUM),CHEKIN(LINNUM),REFLNK,BRANUM,BRALNK(25),
     &  NUMPTH,PATHS(6),FLAG1,FLAG2,X1,X2,ERRFLG,WUNIT2

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &  /AREA20/ERRFLG
     &  /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2

*     ***********
*     * Check each link for it having out-of-range node values.
*     ***********
      DO 10, L=1, NLINK
      IF((NODE1(L).GT.NNODE).OR.(NODE2(L).GT.NNODE))THEN
      IF(ERRFLG.EQ.0) WRITE(5,200)'ERRCHK-I-Link #',
     &  L,'references an out of range node.'
      IF(ERRFLG.EQ.1) WRITE(WUNIT1,200)'ERRCHK-I-Link #',
     &  L,'references an out of range node.'
       IF(ERRFLG.EQ.2) THEN
            WRITE(5,200)'ERRCHK-I-Link #',
     &  L,'references an out of range node.'
            WRITE(WUNIT1,200)'ERRCHK-I-Link #',
     &  L,'references an out of range node.'
       ENDIF
      ENDIF
   10 CONTINUE

*     ***********
*     * Check each link for it having the same node at each end.
*     ***********
      DO 20, L=1, NLINK
      IF(NODE1(L).EQ.NODE2(L))THEN
      IF(ERRFLG.EQ.0) WRITE(5,201)'ERRCHK-I-Link #',
```

```
      &  L,'has the same node at each end.'
    IF(ERRFLG.EQ.1) WRITE(WUNIT1,201)'ERRCHK-I-Link #',
      &  L,'has the same node at each end.'
     IF(ERRFLG.EQ.2) THEN
            WRITE(5,201)'ERRCHK-I-Link #',
      &  L,'has the same node at each end.'
            WRITE(WUNIT2,201)'ERRCHK-I-Link #',
      &  L,'has the same node at each end.'
     ENDIF
    ENDIF
 20 CONTINUE

*    ***********
*    * Check each link for it having the same endpoints as another link.
*    ***********
    DO 30, L1=1, NLINK
    DO 30, L2=1, NLINK
    X=0
    IF(L2.NE.L1)THEN
     IF(NODE1(L1).EQ.NODE1(L2))THEN
     IF(NODE2(L1).EQ.NODE2(L2)) X=1
     ENDIF
     IF(NODE2(L1).EQ.NODE1(L2))THEN
     IF(NODE1(L1).EQ.NODE2(L2)) X=1
     ENDIF
     IF(X.EQ.1)THEN
     IF(ERRFLG.EQ.0) WRITE(5,202)'ERRCHK-I-Link #',
      &  L1,'has the same nodes as link #',L2
     IF(ERRFLG.EQ.1) WRITE(WUNIT1,202)'ERRCHK-I-Link #',
      &  L1,'has the same nodes as link #',L2
      IF(ERRFLG.EQ.2) THEN
            WRITE(5,202)'ERRCHK-I-Link #',
      &  L1,'has the same nodes as link #',L2
            WRITE(WUNIT1,202)'ERRCHK-I-Link #',
      &  L1,'has the same nodes as link #',L2
      ENDIF
     ENDIF
    ENDIF
 30 CONTINUE

*    ***********
*    * Check for continuity of links.
*    ***********

    DO 40, L1=1, NLINK

    IF(CHEKED(L1).EQ.0)THEN

    REFLNK=L1
    FLAG1=0
    BRANUM=0
    CHEKIN(L1)=1

*    *  Find what possible paths there are for the test (initial call).
    CALL PATHS1(NLINK,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
      &  PATHS)
```

```
*     * Search until an exit is found or discontinuity is determined.
      DO WHILE(FLAG1.EQ.0)

      FLAG2=0

*     * Check the endpoints of the link for an exit, and see
*     * if the link has previously been checked off as continuous.
*     * If either condition is met, copy CHEKIN to CHEKED, clear CHEKIN,
*     * and set FLAG1.
      X1=NTYPE(NODE1(REFLNK))
      X2=NTYPE(NODE2(REFLNK))
      IF((X1.LT.0).OR.(X2.LT.0).OR.(CHEKED(REFLNK).EQ.1))THEN
      DO 50,I=1,NLINK
      IF(CHEKIN(I).EQ.1)CHEKED(I)=1
      CHEKIN(I)=0
   50 CONTINUE
      FLAG1=1
      ENDIF

*     * If there is more than one path, increment BRANUM, and
*     * store the identity of the link which lies before the branch.
      IF(NUMPTH.GT.1)THEN
       BRANUM=BRANUM+1
       BRALNK(BRANUM)=REFLNK
      ENDIF

*     * Check to see if all branches and paths have been taken,
*     * and if they have, print a message, clear CHEKIN, and
*     * set FLAG1.
      IF((NUMPTH.EQ.0).AND.(BRANUM.EQ.0).AND.(FLAG1.EQ.0))THEN
       IF(ERRFLG.EQ.0) WRITE(5,203)'ERRCHK-I-Link #',
       & L1,'has no path to an exit.'
       IF(ERRFLG.EQ.1) WRITE(WUNIT1,203)'ERRCHK-I-Link #',
       & L1,'has no path to an exit.'
       IF(ERRFLG.EQ.2) THEN
            WRITE(5,203)'ERRCHK-I-Link #',
       & L1,'has no path to an exit.'
            WRITE(WUNIT1,203)'ERRCHK-I-Link #',
       & L1,'has no path to an exit.'
       ENDIF
       DO 60,I=1,NLINK
       CHEKIN(I)=0
   60  CONTINUE
       FLAG1=1
      ENDIF

*     * If there are no paths, and another branch exists, take it.
*     * Set FLAG2 so further calculations are skipped. The paths for the
*     * branch link are redetermined, and testing can restart from there.
      IF((NUMPTH.EQ.0).AND.(BRANUM.NE.0))THEN
       REFLNK=BRALNK(BRANUM)
       BRANUM=BRANUM-1
       FLAG2=1
      ENDIF

*     * NOTE: from here on, if FLAG2=1, skip calculations.
```

```
*     * Take the first path on the list, if one exists (update REFLNK).
*     * Check the new link off as one which has been passed (update CHEKIN)
      IF((FLAG2.EQ.0).AND.(NUMPTH.NE.0))THEN
       REFLNK=PATHS(1)
       CHEKIN(REFLNK)=1
      ENDIF

*     * Find what possible paths there are for the next test.
      CALL PATHS1(NLINK,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
        & PATHS)

      ENDDO

      ENDIF

   40 CONTINUE

*     ***********
*     * Organize nodes:  higher nodes in NODE1 and lower nodes in NODE2.
*     ***********

      DO 70, L=1, NLINK
      IF(ELEV(NODE2(L)).GT.ELEV(NODE1(L)))THEN
      X=NODE1(L)
      NODE1(L)=NODE2(L)
      NODE2(L)=X
      ENDIF
   70 CONTINUE

  200         FORMAT(1X,A15,I3,1X,A32,/)
  201         FORMAT(1X,A15,I3,1X,A30,/)
  202         FORMAT(1X,A15,I3,1X,A28,I3,/)
  203         FORMAT(1X,A15,I3,1X,A23,/)

      RETURN

      END
```

B.2.13 SUBROUTINE: FRONTAGE

```
      SUBROUTINE FRONTAGE


*****************************************************************
*                                                               *
*                          FRONTAGE                             *
*                                                               *
*****************************************************************
*
*
*
*  FRATIO - ratio uses to calculate the saleable frontage for school &
*           multi-family units
*  TFRONT - total saleable frontage of the sub-division
*  FRONT  - saleable frontage of a given link
*  ASCHOL - total area of school
*  AMULT  - total area of multi-family units
*  ASUB   - total area of the sub-division
*  ACOM   - total area of commerical center
*  AOTHER - total area of others ( industrial area )
*  GROSS  - the gross sub-division area
*  YGROSS - yield efficient based on subdivision's gross area
*  TLEN   - the total street length in the sub-division
*  YSTLEN - yield efficient based on total street length
*
*
*
      PARAMETER NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &     WUNIT1,WUNIT2,RUNIT1,RUNIT2

      COMMON
     &     /AREA3/NLINK
     &     /AREA10/APARK,ABUFF
     &     /AREA25/SPAMAN,SPALIT,SPACAT,FRATIO
     &     /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &     /AREA31/TLEN,TAREA
     &     /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
     &     ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
     &     /AREA40/TFRONT,GROSS,YSTLEN,YGROSS
     &     /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
     &     MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
     &     WKCOST,BRCOST,RLCOST,BLCOST
      FRATIO=177.0
      CONS=10000.0
*
      APAVE=TAREA/CONS
      AWALK=TWK/CONS
*
*  set the total saleable frontage to zero
      TFRONT=0.0
*
*  find the area of single family units
```

```
      ASING=ASUB-APARK-ABUFF-ASCHOL-AMULT-ACOM-AOTHER-APAVE-AWALK
*
*
*     add up the saleable frontage for the single family unit

      write(6,*)nlink
      DO 100 L=1,NLINK
              TFRONT=TFRONT+FRONT(L)
100           CONTINUE
*
      IF ( TFRONT.GT.0.0 )  THEN
*     add the school's frontage to the total
              TFRONT=TFRONT+ASCHOL*FRATIO
*
*     add the multi-family unit's frontage to the total
              TFRONT=TFRONT+AMULT*FRATIO
*
*
*     find the gross area of the subdivision
              GROSS=ASUB-ACOM-AOTHER
*
*
*
*     determine the yield efficient based on total street length
              YSTLEN=TFRONT/TLEN
*
*     determine the yield efficient based on subdivision's gross area
              YGROSS=TFRONT/GROSS
      ELSE
              YGROSS=0.0
              YSTLEN=0.0
              TFRONT=0.0
              GROSS=ASUB-ACOM-AOTHER
      ENDIF
*
*
*
      RETURN
      END
```

B.2.14 SUBROUTINE: HEAD

```
      SUBROUTINE HEAD

              INTEGER ESC,F,BRACK,J,K,P,CODE,SMALLH,QUEST,ZERO,THREE,FOUR,
     2                SMALLL,SEMI,H,SMALLT,B,SMALLC,BS,C,BELL,I
      ESC = 27
      BS = 8
      B = 66
      C = 67
      F = 102
      H = 72
      J = 74
      K = 75
      CODE = 6
      SMALLC = 99
      SMALLH = 104
      SMALLL = 108
      SMALLT = 116
      QUEST = 63
      BRACK = 91
      ZERO = 0
      NUMSGN = 35
      THREE = 51
      FOUR = 52
      SEMI = 59
      BELL = 7

***************
*  HOME CURSOR
***************
      WRITE(6,200)ESC,BRACK,QUEST,CODE,SMALLH
200           FORMAT('+',3A,I1,A)

*************
*  CLEAR SCREEN
*************

      WRITE(6,300)ESC,BRACK,ZERO,J
300           FORMAT('+',2A,I1,A,///)

*************************************
*  PRINT THE HEADER
*************************************

      WRITE(6,400)
      WRITE(6,410)
      WRITE(6,420)
      WRITE(6,430)
      WRITE(6,440)
      WRITE(6,450)
      WRITE(6,460)
      WRITE(6,470)
      WRITE(6,480)
      WRITE(6,490)
      WRITE(6,500)
      WRITE(6,510)
```

```
      WRITE(6,520)
400         FORMAT('1',' ***********     ***********          ********
     2  ***********     *         *')
410         FORMAT(1X,'************   ************        ********
     2 ************   ***       ***')
420         FORMAT(1X,'**          **   **          **       **
     2 **          **   ****    ****')
430         FORMAT(1X,'**          **   **          **      **
     2 **          **   ** **  ** **')
440         FORMAT(1X,'**          **   **          **    **
     2 **          **   **  ****  **')
450         FORMAT(1X,'************   ************   **
     2 ************   **   **   **')
460         FORMAT(1X,'************   ************   **
     2 ************   **        **')
470         FORMAT(1X,'**     **        **          **    **
     2 **          **   **        **')
480         FORMAT(1X,'**      **       **          **      **
     2 **          **   **        **')
490         FORMAT(1X,'**       **      **          **       **
     2 **          **   **        **')
500         FORMAT(1X,'**        **     **          **        ********
     2 **          **   **        **')
510         FORMAT(1X,'**          **   **          **         ********
     2 **          **   **        **')

520         FORMAT(1X,//,5X,'R E S I D E N T I A L   A R E A   C O S T
     2  A N A L Y S I S   M O D E L')

      READ(6,600)
600         FORMAT(A)


      END
```

B.2.15 SUBROUTINE: IDDMSW

```
      SUBROUTINE IDDMSW
*     ***********************************************
*     *  This subroutine identifies which links should  *
*     *   be involved in the domestic sewer system and  *
*     * checks to make sure the links it chooses form a *
*     *                 continuous system.              *
*     ***********************************************
*     * NEWELV=a copy of ELEV for modifications.
*     * OLDELV=a copy of NEWELV for reversing modifications.
*     * CHEKED=keeps track of which links have been cleared.
*     * CHEKIN=keeps track of which links have been passed during
*     *         the test of the current link.
*     * PCHIDX=index into POSCHN.
*     * POSCHN=possible chain currently being checked.
*     * CHNIDX=index into CHAINS.
*     * CHAINS=list of chains which have been found.  Any chains
*     *         which lead to an exit are marked with a one in the
*     *         zero element.
*     * REFNOD=node test is currently referencing.
*     * REFLNK=the link the test is currently referencing - this starts
*     *         at the link being tested, and moves along the path of
*     *         the test.
*     * BRANUM=the number of branches (forks in the path) passed.
*     * BRALNK=holds the link numbers before each branch.
*     * NUMPTH=number of possible paths test can follow.
*     * PATHS =a list of the link numbers of each of the paths.
*     * FLAG1 =indicates how many times the pit routine has been done.
*     * FLAG2 =signifies an end to the test for a given link.
*     * FLAG3 =signifies a return to a former branch location
*     *         and recalculation of possible paths.
*     * PITFLG=is set to indicate a node is in a pit.
*     * LNKTYP=indicates whether a link is coninuous or not
*     *         on the system of obvious links.
*     * RESULT=indicates whether the path choice to be made has
*     *         been made before.
*     * CHECK =used to mark off links which are obviously in the system.
*     * LENGTH=used to calculate the lenth of a chain.
*     * X,X1,X2,X3,I=miscellaneous variables.
*     * V     =used for locating the chain of shortest length.
*     * NDSLNK=number of domestic sewer links.
*     * DLINKS=links involved in domestic sewer system.

      INTEGER
     &  TBTRYW,TBTRYD,TBTRYS,DLINKS,
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,TROAD

      INTEGER CHEKED(150),CHEKIN(150),REFLNK,BRANUM,BRALNK(25),
     &  NUMPTH,PATHS(6),FLAG1,FLAG2,FLAG3,X,X1,X2,X3,ERRFLG,CHECK(150),
     &  PCHIDX,POSCHN(10),CHNIDX,CHAINS(500,0:10),LNKTYP,RESULT,
     &  PITFLG,REFNOD,LINK2,I

      REAL
     &  LLINK,LROAD,ELEV,MULTI,UNITS,V,V1,V2,LENGTH,CHAINL(500),
     &  DSELEV,NEWELV(100),OLDELV(100)
```

```
      COMMON
        &      /AREA2/NNODE,NTYPE(100),ELEV(100)
        &      /AREA3/NLINK,NODE1(150),NODE2(150),LLINK(150),
        &      UNITS(150),MULTI(150),AREA(150)
        &      /AREA4/LROAD(150),WROAD(150),TROAD(150)
        &      /AREA7/NDSLNK,DLINKS(150),NDSGRP,TBTRYD(150,4)
        &   /AREA9/P(150),PDS(150),ASS(150)
        &   /AREA18/RK4(2),DSELEV(100)

*     ***********
*     * Copy (ELEV-2.75) into NEWELV and OLDELV so modifications can be
*     * made to the copy.  The -2.75 is to put the elevations right in
*     * the middle of the 6m trenching range.
*     ***********
      DO 10, L=1, NNODE
      NEWELV(L)=ELEV(L)-2.75
      OLDELV(L)=NEWELV(L)
   10 CONTINUE

*     ***********
*     * Eliminate pits from the system.
*     ***********
  360         DO WHILE(FLAG1.LT.1000)

      DO 20, L1=1, NNODE

      REFNOD=L1

      CALL TAGPIT(NLINK,NODE1,NODE2,NEWELV,REFNOD,PITFLG)

*     * If the node in question is a pit, try to rectify the
*     * problem by lowering the surrounding nodes (only if by
*     * doing so a new pit isn't created).
      IF((PITFLG.EQ.1).AND.(NTYPE(L1).GT.0))THEN

       CALL PATHS5(NLINK,NODE1,NODE2,REFNOD,NUMPTH,PATHS)
       X1=0
       DO 30, L2=1, 4
       X2=PATHS(L2)
       IF(X2.NE.0)THEN
        NEWELV(X2)=NEWELV(REFNOD)-.02
        IF(ELEV(X2)-NEWELV(X2).GT.6.0)NEWELV(X2)=OLDELV(X2)
        CALL TAGPIT(NLINK,NODE1,NODE2,NEWELV,X2,PITFLG)
        IF(PITFLG.EQ.1)NEWELV(X2)=OLDELV(X2)
        IF(OLDELV(X2).NE.NEWELV(X2))X1=X2
       ENDIF
   30  CONTINUE

*     * X1 indicates whether there a pit can be eliminated by
*     * lowering a surrounding node.  Restore all changed elevations
*     * except the one defined by X1.
       DO 40, L2=1, 4
       IF(PATHS(L2).NE.0)THEN
        IF(PATHS(L2).NE.X1)NEWELV(PATHS(L2))=OLDELV(PATHS(L2))
       ENDIF
   40  CONTINUE
```

```
*     * If there are no possibilities, raise REFNOD above the lowest
*     * adjacent node.
      IF(X1.EQ.0)THEN
       V=999999.9
       DO 50, L2=1, 4
        IF(PATHS(L2).NE.0)THEN
         IF((OLDELV(PATHS(L2))-OLDELV(REFNOD)).LT.V)THEN
          V=OLDELV(PATHS(L2))-OLDELV(REFNOD)
          NEWELV(REFNOD)=OLDELV(PATHS(L2))+.02
         ENDIF
        ENDIF
50     CONTINUE
      ENDIF

*     * Update OLDELV.
      DO 60, L=1, NNODE
      OLDELV(L)=(ANINT(NEWELV(L)*100))/100
60    CONTINUE

      ENDIF

20    CONTINUE

      FLAG1=FLAG1+1

      ENDDO

*     ***********
*     * Copy NEWELV into ELEV for evaluations.
*     ***********
      DO 70, L=1, NNODE
      ELEV(L)=(ANINT(NEWELV(L)*100))/100
70    CONTINUE

*     ***********
*     * Organize nodes:  higher nodes in NODE1 and lower nodes in NODE2.
*     ***********

      DO 80, L=1, NLINK
      IF(ELEV(NODE2(L)).GT.ELEV(NODE1(L)))THEN
      X=NODE1(L)
      NODE1(L)=NODE2(L)
      NODE2(L)=X
      ENDIF
80    CONTINUE

*     ***********
*     * Identify links which are most obviously a part of the
*     * domestic sewer system (have a population).  Mark them off
*     * in CHECK.
*     ***********
      DO 90, L=1, NLINK
      V=UNITS(L)+MULTI(L)+PDS(L)
      IF(V.NE.0.0)THEN
       NDSLNK=NDSLNK+1
       DLINKS(NDSLNK)=L
       CHECK(L)=1
```

```
      ENDIF
   90 CONTINUE

*     ***********
*     * Check for continuity of links (those which are obvious).
*     * If a link isn't attached to an exit, scan for possible
*     * connections.  A connection may consist of a single link,
*     * or it may be a chain of many links.  In any case where
*     * there is a choice, take the shortest connection.
*     ***********

      DO 100, L1=1, NDSLNK

      LNKTYP=0

*     ******
*     * Check for continuity.
*     ******
      IF(CHEKED(L1).EQ.0)THEN

       REFLNK=DLINKS(L1)
       FLAG2=0
       BRANUM=0
       CHEKIN(REFLNK)=1

*      * Find what possible paths there are for the test (initial call).
       CALL PATHS2(NDSLNK,DLINKS,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
        &   PATHS)

*      * Delete all upsloping links and any links attached
*      * at the high end of REFLNK.
       DO 110, L2=1,6
       IF(PATHS(L2).NE.0)THEN
        IF(ELEV(NODE2(PATHS(L2))).GE.ELEV(NODE2(REFLNK)))THEN
         PATHS(L2)=0
         NUMPTH=NUMPTH-1
        ENDIF
        X1=NODE1(REFLNK)
        X2=NODE1(PATHS(L2))
        X3=NODE2(PATHS(L2))
        IF((X1.EQ.X2).OR.(X1.EQ.X2))THEN
         PATHS(L2)=0
         NUMPTH=NUMPTH-1
        ENDIF
       ENDIF
  110          CONTINUE
       DO 120, L2=1, 5
       DO 130, L3=1, 5
       IF(PATHS(L3).EQ.0)THEN
        DO 140, L4=L3, 5
        PATHS(L4)=PATHS(L4+1)
        PATHS(L4+1)=0
  140           CONTINUE
       ENDIF
  130          CONTINUE
  120          CONTINUE
```

```
*     * Search until an exit is found or discontinuity is determined.
      DO WHILE(FLAG2.EQ.0)

      FLAG3=0

*     ***
*     * Check the endpoints of the link for an exit, and see
*     * if the link has previously been checked off as continuous.
*     * If either condition is met, copy CHEKIN to CHEKED, clear CHEKIN,
*     * and set FLAG2.
*     ***
      X1=NTYPE(NODE1(REFLNK))
      X2=NTYPE(NODE2(REFLNK))
      IF((X1.LT.0).OR.(X2.LT.0).OR.(CHEKED(REFLNK).EQ.1))THEN
       DO 150,I=1,NDSLNK
       IF(CHEKIN(DLINKS(I)).EQ.1)CHEKED(DLINKS(I))=1
       CHEKIN(I)=0
150          CONTINUE
       FLAG2=1
      ENDIF

*     ***
*     * If there is more than one path, increment BRANUM, and
*     * store the identity of the link which lies before the branch.
*     ***
      IF(NUMPTH.GT.1)THEN
       BRANUM=BRANUM+1
       BRALNK(BRANUM)=REFLNK
      ENDIF

*     ***
*     * Check to see if all branches and paths have been taken,
*     * and if they have, set LNKTYP to indicate no path, clear CHEKIN,
*     * and set FLAG2.
*     ***
      IF((NUMPTH.EQ.0).AND.(BRANUM.EQ.0).AND.(FLAG2.EQ.0))THEN
       DO 160,I=1,NLINK
       CHEKIN(I)=0
160          CONTINUE
       LNKTYP=1
       FLAG2=1
      ENDIF

*     ***
*     * If there are no paths, and another branch exists, take it.
*     * Set FLAG3 so further calculations are skipped.  The paths for the
*     * branch link are redetermined, and testing can restart from there.
*     ***
      IF((NUMPTH.EQ.0).AND.(BRANUM.NE.0))THEN
       REFLNK=BRALNK(BRANUM)
       BRANUM=BRANUM-1
       FLAG3=1
      ENDIF

*     *  NOTE: from here on, if FLAG3=1, skip calculations.

*     ***
```

```
*     * Take the first path on the list, if one exists (update REFLNK).
*     * Check the new link off as one which has been passed (update CHEKIN).
*     ***
      IF((FLAG3.EQ.0).AND.(NUMPTH.NE.0))THEN
       REFLNK=PATHS(1)
       CHEKIN(REFLNK)=1
      ENDIF

*     ***
*     * Find what possible paths there are for the next test.
*     ***
     CALL PATHS2(NDSLNK,DLINKS,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
       &  PATHS)

*     * Delete all upsloping links and any links attached
*     * at the high end of REFLNK.
      DO 170, L2=1,6
      IF(PATHS(L2).NE.0)THEN
       IF(ELEV(NODE2(PATHS(L2))).GE.ELEV(NODE2(REFLNK)))THEN
        PATHS(L2)=0
        NUMPTH=NUMPTH-1
       ENDIF
       X1=NODE1(REFLNK)
       X2=NODE1(PATHS(L2))
       X3=NODE2(PATHS(L2))
       IF((X1.EQ.X2).OR.(X1.EQ.X2))THEN
        PATHS(L2)=0
        NUMPTH=NUMPTH-1
       ENDIF
      ENDIF
170          CONTINUE
      DO 180, L2=1, 5
      DO 190, L3=1, 5
      IF(PATHS(L3).EQ.0)THEN
       DO 200, L4=L3, 5
       PATHS(L4)=PATHS(L4+1)
       PATHS(L4+1)=0
200           CONTINUE
      ENDIF
190          CONTINUE
180          CONTINUE

      ENDDO

     ENDIF

*     ******
*     * If the above routine determines that a link of the system
*     * isn't somehow linked to an exit, this routine creates
*     * possible paths and picks the shortest one, adding to the system
*     * the links it uses in creating the path.
*     ******
     IF(LNKTYP.EQ.1)THEN

      REFLNK=DLINKS(L1)
      PCHIDX=1
      POSCHN(PCHIDX)=REFLNK
```

```
      CHNIDX=0
      FLAG2=0
      BRANUM=0
      CHEKIN(REFLNK)=1

*     * Find what possible paths there are for the test (initial call).
     CALL PATHS4(NLINK,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
       &  PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)

*     * Search until all chains have been determined.
      DO WHILE(FLAG2.EQ.0)

      FLAG3=0

*     ***
*     * Check the endpoints of the link for an exit.
*     * If the condition is met, increment CHNIDX, copy POSCHN
*     * to CHAINS, pull a branch ID from BRALNK, erase POSCHN backwards
*     * till it hits the ID (so from the branch on, the chain is new),
*     * set FLAG3 to idicate a skip of further calculations so that
*     * a new chain can be developed from the branch on.  If there aren't
*     * any branches left, set FLAG2.
*     ***
      X1=NTYPE(NODE1(REFLNK))
      X2=NTYPE(NODE2(REFLNK))

*     * Check for an exit.
      IF((X1.LT.0).OR.(X2.LT.0))THEN

*     * Increment CHNIDX and copy POSCHN into CHAINS.  Set the
*     * zero element to 1, since this chain reaches an exit.
       CHNIDX=CHNIDX+1
       DO 210, L2=1, PCHIDX
       CHAINS(CHNIDX,L2)=POSCHN(L2)
210            CONTINUE
       CHAINS(CHNIDX,0)=1

*     * Set FLAG2 if there aren't any branches left.
       IF(BRANUM.EQ.0) FLAG2=1

*     * If there are branches left, get the last branch and erase
*     * POSCHN back until that branch location (so a new chain can
*     * be built from there on).  Also, clear CHEKIN elements
*     * of erased POSCHN links.
       IF(BRANUM.NE.0)THEN
        REFLNK=BRALNK(BRANUM)
        BRANUM=BRANUM-1
        DO WHILE(POSCHN(PCHIDX).NE.REFLNK)
        CHEKIN(POSCHN(PCHIDX))=0
        PCHIDX=PCHIDX-1
        ENDDO
       ENDIF

*     * Set FLAG3 so further calculations are skipped;  start
*     * generating a new chain from the branch.
       FLAG3=1
```

```
      ENDIF

*     ***
*     * If there is more than one path, increment BRANUM, and
*     * store the identity of the link which lies before the branch.
*     ***
      IF(NUMPTH.GT.1)THEN
       BRANUM=BRANUM+1
       BRALNK(BRANUM)=REFLNK
      ENDIF

*     * NOTE: from here on, if FLAG3=1, skip calculations.

*     ***
*     * If there are no paths, and another branch exists, take it.
*     * Set FLAG3 so further calculations are skipped.  The paths for the
*     * branch link are redetermined, and testing can restart from there.
*     * Increment CHNIDX and copy POSCHN into CHAINS, and erase POSCHN
*     * back to the branch (erasing CHEKIN entries along the way).
*     ***
      IF((FLAG3.EQ.0).AND.(NUMPTH.EQ.0))THEN

*     * Increment CHNIDX and copy POSCHN into CHAINS.
       CHNIDX=CHNIDX+1
       DO 220, L2=1, PCHIDX
       CHAINS(CHNIDX,L2)=POSCHN(L2)
220          CONTINUE

*     * Set FLAG2 if there aren't any branches left.
       IF(BRANUM.EQ.0) FLAG2=1

*     * If there are branches left, get the last branch and erase
*     * POSCHN back until that branch location (so a new chain can
*     * be built from there on).  Also, clear CHEKIN elements of
*     * erased POSCHN links.
       IF(BRANUM.NE.0)THEN
        REFLNK=BRALNK(BRANUM)
        BRANUM=BRANUM-1
        DO WHILE(POSCHN(PCHIDX).NE.REFLNK)
        CHEKIN(POSCHN(PCHIDX))=0
        PCHIDX=PCHIDX-1
        ENDDO
       ENDIF

*     * Set FLAG3 so further calculations are skipped;  start
*     * generating a new chain from the branch.
       FLAG3=1

      ENDIF

*     ***
*     * Take the first path on the list, if one exists (update REFLNK).
*     * Check the new link off as one which has been passed (update CHEKIN).
*     * Also update POSCHN.
*     ***
      IF((FLAG3.EQ.0).AND.(NUMPTH.NE.0))THEN
       REFLNK=PATHS(1)
```

```
      CHEKIN(REFLNK)=1
      PCHIDX=PCHIDX+1
      POSCHN(PCHIDX)=REFLNK
     ENDIF

*     ***
*     * Find what possible paths there are for the next test.
*     ***
    CALL PATHS4(NLINK,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
      &  PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)

     ENDDO

*     ******
*     * Pick the shortest path and copy all its elements into
*     * CHEKED.
*     ******

*     ***
*     * Calculate the lengths for each chain which ends in an exit.
*     * Store the results in CHAINL.
*     ***
     DO 230, L2=1, CHNIDX
     LENGTH=0.0
     IF(CHAINS(L2,0).EQ.1)THEN
      X1=2
      DO WHILE((CHAINS(L2,X1).NE.0).AND.(X1.LE.10))
      LENGTH=LENGTH+LLINK(CHAINS(L2,X1))
      X1=X1+1
      ENDDO
     ENDIF
     CHAINL(L2)=LENGTH
230          CONTINUE

*     ***
*     * Search for a nonzero chain length.
*     ***
     X2=0
     DO 240, L2=1, 500
     IF(CHAINL(L2).NE.0.0)X2=L2
240          CONTINUE

*     ***
*     * Search for the shortest chain.
*     ***
     DO 250, L2=1, 500
     IF((CHAINL(L2).LT.CHAINL(X2)).AND.(CHAINL(L2).NE.0.0))X2=L2
250          CONTINUE

*     ***
*     * Mark off the links involved in the chain in CHECK.
*     ***
     DO 260, L2=1, 10
     CHECK(CHAINS(X2,L2))=1
260          CONTINUE

*     ***
```

```
*     * Clear CHAINS.
*     ***
   DO 270, L2=1, 500
   DO 280, L3=0, 10
   CHAINS(L2,L3)=0
280         CONTINUE
270         CONTINUE

*     ***
*     * Clear CHAINL.
*     ***
   DO 290, L2=1, 500
   CHAINL(L2)=0.0
290         CONTINUE

*     ***
*     * Clear CHEKIN.
*     ***
   DO 300, L2=1, 150
   CHEKIN(L2)=0
300         CONTINUE

   ENDIF

*     ***********
*     * Update DLINKS.
*     ***********
   X=0
   DO 310, L=1, 150
   IF(CHECK(L).NE.0)THEN
   X=X+1
   DLINKS(X)=L
   ENDIF
310         CONTINUE
   NDSLNK=X

100         CONTINUE

*     ***********
*     * Eliminate level links.  Pick the end with the shortest
*     * path out and lower it (or raise the other end if lowering
*     * the first causes a pit to be formed).
*     ***********
   X=0
   DO 320, L1=1, NLINK
   IF(ELEV(NODE1(L1)).EQ.ELEV(NODE2(L1)))THEN
    CALL PATHS7(NODE1,NODE2,NODE1(L1),NDSLNK,DLINKS,PATHS)
    V1=999999.9
    DO 330, L2=1,4
    IF(PATHS(L2).NE.0)THEN
     CALL OPTPTH(NDSLNK,DLINKS,PATHS(L2),V)
     IF(V.LT.V1)V1=V
    ENDIF
330          CONTINUE
    CALL PATHS7(NODE1,NODE2,NODE2(L1),NDSLNK,DLINKS,PATHS)
    V2=999999.9
    DO 340, L2=1,4
```

```
      IF(PATHS(L2).NE.0)THEN
       CALL OPTPTH(NDSLNK,DLINKS,PATHS(L2),V)
       IF(V2.LT.V)V2=V
      ENDIF
340          CONTINUE
      IF(V1.LT.V2)THEN
       ELEV(NODE1(L1))=ELEV(NODE1(L1))-.01
       CALL TAGPIT(NLINK,NODE1,NODE2,ELEV,NODE1(L1),PITFLG)
       IF(PITFLG.EQ.1)ELEV(NODE1(L1))=ELEV(NODE1(L1))+.01
       IF(PITFLG.EQ.1)ELEV(NODE2(L1))=ELEV(NODE2(L1))+.01
       IF(PITFLG.EQ.1)X=1
      ENDIF
      IF(V2.LT.V1)THEN
       ELEV(NODE2(L1))=ELEV(NODE2(L1))-.01
       CALL TAGPIT(NLINK,NODE1,NODE2,ELEV,NODE2(L1),PITFLG)
       IF(PITFLG.EQ.1)ELEV(NODE2(L1))=ELEV(NODE2(L1))+.01
       IF(PITFLG.EQ.1)ELEV(NODE1(L1))=ELEV(NODE1(L1))+.01
       IF(PITFLG.EQ.1)X=X+1
      ENDIF
     ENDIF
320        CONTINUE

     IF(X.EQ.1)THEN
      DO 350, L=1, NNODE
      NEWELV(L)=ELEV(L)
      OLDELV(L)=ELEV(L)
350          CONTINUE
      FLAG1=0
     ENDIF
     IF(X.EQ.1)GOTO360

     RETURN

     END
```

B.2.16 SUBROUTINE: IDSTSW

```
      SUBROUTINE IDSTSW
*       ***************************************************
*       *  This subroutine identifies which links should  *
*       *      be involved in the domestic sewer system.  *
*       ***************************************************
*       * NSSLNK=number of watermain links.
*       * SLINKS=links involved in watermain system.

      INTEGER NLINK,NSSLNK,SLINKS

      COMMON
       &      /AREA3/NLINK,NODE1(150),NODE2(150),LLINK(150),
       &      /AREA8/NSSLNK,SLINKS(150),NSSGRP,TBTRYS(150,4)

*       ***********
*       * Identify links which a part of the storm sewer system.
*       ***********
      DO 10, L=1, NLINK
      SLINKS(L)=L
   10 CONTINUE
      NSSLNK=NLINK

      RETURN

      END
```

B.2.17 SUBROUTINE: IDWATR

```
      SUBROUTINE IDWATR
*     ***********************************************
*     * This subroutine identifies which links should  *
*     * be involved in the watermain system and checks *
*     *     to make sure the links it chooses form a    *
*     *                continuous system.               *
*     ***********************************************
*     * CHEKED=keeps track of which links have been cleared.
*     * CHEKIN=keeps track of which links have been passed during
*     *        the test of the current link.
*     * PCHIDX=index into POSCHN.
*     * POSCHN=possible chain currently being checked.
*     * CHNIDX=index into CHAINS.
*     * CHAINS=list of chains which have been found.  Any chains
*     *        which lead to an exit are marked with a one in the
*     *        zero element.
*     * REFLNK=the link the test is currently referencing - this starts
*     *        at the link being tested, and moves along the path of
*     *        the test.
*     * BRANUM=the number of branches (forks in the path) passed.
*     * BRALNK=holds the link numbers before each branch.
*     * NUMPTH=number of possible paths test can follow.
*     * PATHS =a list of the link numbers of each of the paths.
*     * FLAG1 =signifies an end to the test for a given link.
*     * FLAG2 =signifies a return to a former branch location
*     *        and recalculation of possible paths.
*     * LNKTYP=indicates whether a link is coninuous or not
*     *        on the system of obvious links.
*     * RESULT=indicates whether the path choice to be made has
*     *        been made before.
*     * CHECK =used to mark off links which are obviously in the system.
*     * LENGTH=used to calculate the lenth of a chain.
*     * X,X1,X2=miscellaneous variables.
*     * V     =used for locating the chain of shortest length.
*     * NWMLNK=number of watermain links.
*     * WLINKS=links involved in watermain system.

      INTEGER
     &  TBTRYW,TBTRYD,TBTRYS,WLINKS,
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,TROAD

      INTEGER CHEKED(150),CHEKIN(150),REFLNK,BRANUM,BRALNK(25),
     &  NUMPTH,PATHS(6),FLAG1,FLAG2,X1,X2,ERRFLG,CHECK(150),
     &  PCHIDX,POSCHN(10),CHNIDX,CHAINS(500,0:10),LNKTYP,RESULT,QQ,QQQ

      REAL
     &  LLINK,LROAD,WROAD,MULTI,UNITS,V,LENGTH,CHAINL(500)

      COMMON
     &     /AREA2/NNODE,NTYPE(100),ELEV(100)
     &     /AREA3/NLINK,NODE1(150),NODE2(150),LLINK(150),
     &     UNITS(150),MULTI(150),AREA(150)
     &     /AREA4/LROAD(150),WROAD(150),TROAD(150)
     &     /AREA6/NWMLNK,WLINKS(150),NWMGRP,TBTRYW(150,4)
     &  /AREA9/QQ,QQQ,P(150),PDS(150),ASS(150)
```

```
      NWMLNK=0

*     ***********
*     * Identify links which are most obviously a part of the
*     * watermain system (have a population).  Mark them off
*     * in CHECK.
*     ***********
      DO 10, L=1, NLINK
      V=UNITS(L)+MULTI(L)+P(L)
      IF(V.NE.0.0)THEN
       NWMLNK=NWMLNK+1
       WLINKS(NWMLNK)=L
       CHECK(L)=1
      ENDIF
   10 CONTINUE

*     ***********
*     * Check for continuity of links (those which are obvious).
*     * If a link isn't attached to an exit, scan for possible
*     * connections.  A connection may consist of a single link,
*     * or it may be a chain of many links.  In any case where
*     * there is a choice, take the shortest connection.
*     ***********

      DO 20, L1=1, NWMLNK

*     ******
*     * Check for continuity.
*     ******
      IF(CHEKED(L1).EQ.0)THEN

       REFLNK=WLINKS(L1)
       FLAG1=0
       LNKTYP=0
       BRANUM=0
       CHEKIN(REFLNK)=1

*      *  Find what possible paths there are for the test (initial call).
       CALL PATHS2(NWMLNK,WLINKS,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
        &   PATHS)

*      *  Search until an exit is found or discontinuity is determined.
       DO WHILE(FLAG1.EQ.0)

       FLAG2=0

*      ***
*      * Check the endpoints of the link for an exit, and see
*      * if the link has previously been checked off as continuous.
*      * If either condition is met, copy CHEKIN to CHEKED, clear CHEKIN,
*      * and set FLAG1.
*      ***
       X1=NTYPE(NODE1(REFLNK))
       X2=NTYPE(NODE2(REFLNK))
       IF((X1.LT.0).OR.(X2.LT.0).OR.(CHEKED(REFLNK).EQ.1))THEN
        DO 30,I=1,NLINK
```

```
      IF(CHEKIN(I).EQ.1)CHEKED(I)=1
      CHEKIN(I)=0
30    CONTINUE
      FLAG1=1
     ENDIF

*     ***
*     * If there is more than one path, increment BRANUM, and
*     * store the identity of the link which lies before the branch.
*     ***
     IF(NUMPTH.GT.1)THEN
      BRANUM=BRANUM+1
      BRALNK(BRANUM)=REFLNK
     ENDIF

*     ***
*     * Check to see if all branches and paths have been taken,
*     * and if they have, set LNKTYP to indicate no path, clear CHEKIN,
*     * and set FLAG1.
*     ***
     IF((NUMPTH.EQ.0).AND.(BRANUM.EQ.0).AND.(FLAG1.NE.1))THEN
      DO 40,I=1,NLINK
      CHEKIN(I)=0
40    CONTINUE
      LNKTYP=1
      FLAG1=1
     ENDIF

*     ***
*     * If there are no paths, and another branch exists, take it.
*     * Set FLAG2 so further calculations are skipped. The paths for the
*     * branch link are redetermined, and testing can restart from there.
*     ***
     IF((NUMPTH.EQ.0).AND.(BRANUM.NE.0))THEN
      REFLNK=BRALNK(BRANUM)
      BRANUM=BRANUM-1
      FLAG2=1
     ENDIF

*     * NOTE: from here on, if FLAG2=1, skip calculations.

*     ***
*     * Take the first path on the list, if one exists (update REFLNK).
*     * Check the new link off as one which has been passed (update CHEKIN)
*     ***
     IF((FLAG1.EQ.0).AND.(FLAG2.EQ.0).AND.(NUMPTH.NE.0))THEN
      REFLNK=PATHS(1)
      CHEKIN(REFLNK)=1
     ENDIF

*     ***
*     * Find what possible paths there are for the next test.
*     ***
    CALL PATHS2(NWMLNK,WLINKS,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
     & PATHS)

     ENDDO
```

```
      ENDIF

*       ******
*       * If the above routine determines that a link of the system
*       * isn't somehow linked to an exit, this routine creates
*       * possible paths and picks the shortest one, adding the links
*       * it uses in creating the path, to the system.
*       ******

*       * This line was inserted to force a priority on the watermain system.
*       * This priority is described in the Programmer's section of the manual
*       * under the description of IDWATR.
      LNKTYP=1

      IF(LNKTYP.EQ.1)THEN

*       * Mark off dead ends.
       DO 50, L2=1, NWMLNK
       IF(NTYPE(NODE1(WLINKS(L2))).EQ.1)CHEKIN(WLINKS(L2))=1
       IF(NTYPE(NODE2(WLINKS(L2))).EQ.1)CHEKIN(WLINKS(L2))=1
50     CONTINUE

       REFLNK=WLINKS(L1)
       PCHIDX=1
       POSCHN(PCHIDX)=REFLNK
       CHNIDX=0
       FLAG1=0
       BRANUM=0
       CHEKIN(REFLNK)=1

*       *  Find what possible paths there are for the test (initial call).
      CALL PATHS3(NLINK,NODE1,NODE2,WROAD,REFLNK,CHEKIN,NUMPTH,
       &  PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)

*       *  Search until all chains have been determined.
       DO WHILE(FLAG1.EQ.0)

       FLAG2=0

*       ***
*       * Check the endpoints of the link for an exit.
*       * If the condition is met, increment CHNIDX, copy POSCHN
*       * to CHAINS, pull a branch ID from BRALNK, erase POSCHN backwards
*       * till it hits the ID (so from the branch on, the chain is new),
*       * set FLAG2 to idicate a skip of further calculations so that
*       * a new chain can be developed from the branch on.  If there aren't
*       * any branches left, set FLAG1.
*       ***
       X1=NTYPE(NODE1(REFLNK))
       X2=NTYPE(NODE2(REFLNK))

*       * Check for an exit.
       IF((X1.LT.0).OR.(X2.LT.0))THEN

*       * Increment CHNIDX and copy POSCHN into CHAINS.  Set the
*       * zero element to 1, since this chain reaches an exit.
```

```
      CHNIDX=CHNIDX+1
      DO 60, L2=1, PCHIDX
      CHAINS(CHNIDX,L2)=POSCHN(L2)
60    CONTINUE
      CHAINS(CHNIDX,0)=1

*     * Set FLAG1 if there aren't any branches left.
      IF(BRANUM.EQ.0) FLAG1=1

*     * If there are branches left, get the last branch and erase
*     * POSCHN back until that branch location (so a new chain can
*     * be built from there on).  Also, clear CHEKIN elements
*     * of erased POSCHN links.
      IF(BRANUM.NE.0)THEN
       REFLNK=BRALNK(BRANUM)
       BRANUM=BRANUM-1
       DO WHILE(POSCHN(PCHIDX).NE.REFLNK)
       CHEKIN(POSCHN(PCHIDX))=0
       PCHIDX=PCHIDX-1
       ENDDO
      ENDIF

*     * Set FLAG2 so further calculations are skipped;  start
*     * generating a new chain from the branch.
      FLAG2=1

     ENDIF

*     ***
*     * If there is more than one path, increment BRANUM, and
*     * store the identity of the link which lies before the branch.
*     ***
     IF(NUMPTH.GT.1)THEN
      BRANUM=BRANUM+1
      BRALNK(BRANUM)=REFLNK
     ENDIF

*     *  NOTE: from here on, if FLAG2=1, skip calculations.

*     ***
*     * If there are no paths, and another branch exists, take it.
*     * Set FLAG2 so further calculations are skipped.  The paths for the
*     * branch link are redetermined, and testing can restart from there.
*     * Increment CHNIDX and copy POSCHN into CHAINS, and erase POSCHN
*     * back to the branch (erasing CHEKIN entries along the way).
*     ***
     IF((FLAG2.EQ.0).AND.(NUMPTH.EQ.0))THEN

*     * Increment CHNIDX and copy POSCHN into CHAINS.
      CHNIDX=CHNIDX+1
      DO 70, L2=1, PCHIDX
      CHAINS(CHNIDX,L2)=POSCHN(L2)
70    CONTINUE

*     * Set FLAG1 if there aren't any branches left.
      IF(BRANUM.EQ.0) FLAG1=1
```

```
*     * If there are branches left, get the last branch and erase
*     * POSCHN back until that branch location (so a new chain can
*     * be built from there on).  Also, clear CHEKIN elements of
*     * erased POSCHN links.
      IF(BRANUM.NE.0)THEN
       REFLNK=BRALNK(BRANUM)
       BRANUM=BRANUM-1
       DO WHILE(POSCHN(PCHIDX).NE.REFLNK)
       CHEKIN(POSCHN(PCHIDX))=0
       PCHIDX=PCHIDX-1
       ENDDO
      ENDIF

*     * Set FLAG2 so further calculations are skipped;  start
*     * generating a new chain from the branch.
      FLAG2=1

     ENDIF

*     ***
*     * Take the first path on the list, if one exists (update REFLNK).
*     * Check the new link off as one which has been passed (update CHEKIN).
*     * Also update POSCHN.
*     ***

     IF((FLAG2.EQ.0).AND.(NUMPTH.NE.0))THEN
      REFLNK=PATHS(1)
      CHEKIN(REFLNK)=1
      PCHIDX=PCHIDX+1
      POSCHN(PCHIDX)=REFLNK
     ENDIF

*     ***
*     *  Find what possible paths there are for the next test.
*     ***
    CALL PATHS3(NLINK,NODE1,NODE2,WROAD,REFLNK,CHEKIN,NUMPTH,
     &  PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)

     ENDDO

*     ******
*     * Pick the shortest path and copy all its elements into
*     * CHEKED.
*     ******

*     ***
*     * Calculate the lengths for each chain which ends in an exit.
*     * Store the results in CHAINL.
*     ***
     DO 80, L2=1, CHNIDX
     LENGTH=0.0
     IF(CHAINS(L2,0).EQ.1)THEN
      X1=2
      DO WHILE((CHAINS(L2,X1).NE.0).AND.(X1.LE.10))
      LENGTH=LENGTH+LLINK(CHAINS(L2,X1))
      X1=X1+1
      ENDDO
```

```
      ENDIF
      CHAINL(L2)=LENGTH
80    CONTINUE

*      ***
*      * Search for a nonzero chain length.
*      ***
      X2=0
      DO 90, L2=1, 500
      IF(CHAINL(L2).NE.0.0)X2=L2
90    CONTINUE

*      ***
*      * Search for the shortest chain.
*      ***
      DO 100, L2=1, 500
      IF((CHAINL(L2).LT.CHAINL(X2)).AND.(CHAINL(L2).NE.0.0))X2=L2
100           CONTINUE

*      ***
*      * Mark off the links involved in the chain in CHECK.
*      ***
      DO 110, L2=1, 10
      CHECK(CHAINS(X2,L2))=1
110           CONTINUE

*      ***
*      * Clear CHAINS.
*      ***
     DO 120, L2=1, 500
     DO 130, L3=0, 10
     CHAINS(L2,L3)=0
130          CONTINUE
120          CONTINUE

*      ***
*      * Clear CHAINL.
*      ***
     DO 140, L2=1, 500
     CHAINL(L2)=0.0
140          CONTINUE

     ENDIF

*      ***
*      * Clear CHEKIN.
*      ***
     DO 150, L2=1, 150
     CHEKIN(L2)=0
150          CONTINUE

20 CONTINUE

*      ***********
*      * Update WLINKS.
*      ***********
     X=0
```

```
      DO 160, L=1, 150
      IF(CHECK(L).NE.0)THEN
      X=X+1
      WLINKS(X)=L
      ENDIF
160         CONTINUE
      NWMLNK=X

      RETURN

      END
```

B.2.18 SUBROUTINE: INDAT1

```
   SUBROUTINE INDAT1
*****************************************************************
*                                                               *
*                      SUBROUTINE INDAT1                        *
*                                                               *
*          This subroutine READS the Unit Cost Data that        *
*          pertains to the study area.                          *
*                                                               *
*****************************************************************
*
*
*    GLOBAL VARIABLES
*
*  RUNIT1   - THE NUMBER OF THE DEVICE THAT READS THE UNIT
*                 COST DATA (This number will usually be a 4).
*  COST1    - STREET SIGN ($/SIGN)
*  COST2    - STREET LIGHT ($/LIGHT)
*  COST3    - PARK DEVELOPMENT ($/HECTARE)
*  COST4    - BUFFER DEVELOPMENT ($/HECTARE)
*  COST5    - COLLECTOR STREET ($/SQ.M)
*  COST6    - LOCAL STREET ($/SQ.M)
*  COST7    - SIDEWALK ($/SQ.M)
*  COST8    - BARRIER CURB & GUTTER ($/M)
*  COST9    - ROLLED CURB & GUTTER ($/M)
*  COST10   - BLVD CURB & GUTTER ($/M)
*  COST11   - WATERMAIN ($/M)
*  COST12   - DOMESTIC SEWER ($/M)
*  COST13   - STORM SEWER ($/M)
*  COST14   - MANHOLE - BASE, COVER, RISER, FRAME ($/MANHOLE)
*  COST15   - MANHOLE SHAFT ($/M-DEPTH)
*  COST16   - CATCHBASIN - FRAME, COVER, ETC ($/CATCHBASIN)
*  COST17   - CATCHBASIN SHAFT ($/M-DEPTH)
*  COST18   - CATCHBASIN LEAD ($/M)
*  COST19   - POWER - SINGLE FAMILY ($/UNIT)
*  COST20   - POWER - MULTI  FAMILY ($/UNIT)
*  COST21   - GAS   - SINGLE FAMILY ($/UNIT)
*  COST22   - GAS   - MULTI  FAMILY ($/UNIT)
*  COST23   - PHONE - SINGLE FAMILY ($/UNIT)
*  COST24   - PHONE - MULTI  FAMILY ($/UNIT)
*  COST25   - STREET SWEEPING ($/KM/YR)
*  COST26   - WINTER ROAD ($/KM/YR)
*  COST27   - ASPHALT MAINTENANCE ($/KM/YR)
*  COST28   - CONCRETE MAINTENANCE ($/KM/YR)
*  COST29   - SEWER MAINTENANCE ($/KM/YR)
*  COST30   - WATER DISTRIBUTION ($/KM/YR)
*  COST31   - SOLID WASTE COLLECTION ($/HECTARE/YR)
*  COST32   - STREET LIGHT MAINTENANCE ($/LIGHT/YR)
*  COST33   - PARK & BUFFER MAINTENANCE ($/HECTARE/YR)
*
*    LOCAL VARIABLES
*
*  I,J      - LOOP CONTROL VARIABLES
*       D1-4        - (-1) FLAGS INDICATING END OF DATA SECTION
*      ACCODE       - CODE CORRESPONDING TO AN ACCOUNT NAME AND NUMBER.
*             ACCODE=5 FOR ACCNUM(5) AND DEFINE(5).)
```

```
*       NUMELE        - NUMBER OF SUBDIVISION ELEMENTS.
*       I,J        - LOOP CONTROL VARIABLES
*       L,N        - LOOP CONTROL VARIABLES
*                    where: L refers to the link number.
*                           N refers to the node number.
*
*
      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &      RUNIT1,RUNIT2,WUNIT1,WUNIT2,IST,ACCODE,OLIFE

      REAL MBCOST,MINST,MGRAD,INTRST

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*35,DEFINE

      COMMON
        &      /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
        &      /AREA11/COST1,COST2/AREA12/COST3,COST4/AREA13/COST5,COST6
        &      /AREA14/COST7,COST8,COST9,COST10
        &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
        &   COST17,COST18(9)
        &      /AREA16/COST19,COST20,COST21,COST22,COST23,COST24
        &      /AREA17/COST25,COST26,COST27,COST28,COST29,COST30,COST31,
        &   COST32,COST33,MBCOST(25),OBCOST(25)
        &      /AREA20/ID1FLG,ID2FLG,OD1FLG,OD2FLG,WATFLG,DMSFLG,STMFLG,MANFLG,
        &      SDWFLG,STRFLG,STLFLG,STSFLG,UTIFLG,PARFLG,OPRFLG,ECOFLG,ERRFLG
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA22/PEOPLE,WVMIN,FRICTN,USE,FLOAD
        &      /AREA23/DSCOVR,DSVMIN,DSVMAX
        &      /AREA24/SSCOVR,SSVMIN,SSVMAX
        &      /AREA25/SPAMAN,SPALIT
        &      /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
        &      MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
        &      WKCOST,BRCOST,RLCOST,BLCOST
        &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10),
        &      MNPACC(25,10),OPPACC(25,10)
        &      /AREA47/NACCNT,ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100),
        &      DEFINE(100)
        &      /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &      ,MNCOST(25,3),MCAPAF(25,10,3)
        &      /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &      ,OPCOST(25,3),OCAPAF(25,10,3)
      PRINT*
      PRINT*
      PRINT*,'E C O N O M I C   D A T A'
      PRINT*
      IST=1
      CALL IOFILE (RUNIT1,IST)
      PRINT*
      PRINT*,'READING ECONOMIC DATA'
*
*****************************************************************************
*
*  Reads the interest rate and subdivision life
```

```
*          values. The following sections all stop
*          reading their respective data when the
*          data contains either 99 or -1.
*
*****************************************************************
*
   DO WHILE (INTRST(N).NE.-1)
            N=N+1
            READ(RUNIT1,80)INTRST(N),LIFE(N)
   END DO
   IF (INTRST(N).EQ.-1)THEN
            N=N-1
            NINT=N
   ENDIF
*
*****************************************************************
*           This portion of the routine reads the service       *
*           element's unit costs.                               *
*****************************************************************
*
   READ(RUNIT1,40)COST1,COST2,COST3,COST4,COST5,COST6,COST7,
     &      COST8,COST9,COST10
   READ(RUNIT1,42)(COST11(I),I=1,6)
   READ(RUNIT1,44)((COST12(I,J),I=1,9),J=1,5)
   READ(RUNIT1,44)((COST13(I,J),I=1,9),J=1,14)
   READ(RUNIT1,40)COST14,COST15,COST16,COST17
   READ(RUNIT1,44)(COST18(I),I=1,9)
   READ(RUNIT1,40)COST19,COST20,COST21,COST22,COST23,COST24,
     &  COST25,COST26
        READ(RUNIT1,40)COST27,COST28,COST29,COST30,COST31,
     &  COST32,COST33
*****************************************************************
*       Read the various calibration data values.               *
*****************************************************************
   READ(RUNIT1,45)PEOPLE,WVMIN,FRICTN,USE,FLOAD
   READ(RUNIT1,45)DSCOVR,DSVMIN,DSVMAX
   READ(RUNIT1,45)SSCOVR,SSVMIN,SSVMAX
   READ(RUNIT1,45)SPAMAN,SPALIT
*****************************************************************
*       Calculate the total number of accouts, NACCNT.Read the accout numbe
*  ACCNUM,
*       and corresponding account names, DEFINE, from the link/node data fi
*****************************************************************
   N=0
   DO WHILE (ACCNUM(N).NE.'          -1')
            N=N+1
            READ(RUNIT1,175)ACCNUM(N),DEFINE(N)
   END DO
   IF (ACCNUM(N).EQ.'          -1')THEN
            N=N-1
            NACCNT=N
   ENDIF
*****************************************************************
*       Read the number of accounts associated with each subdivision elemen
*       NACONT.
*       Read the account code, ACCODE, and the cost percentage of each
*       subdivision element by account, PRCENT.  Assign the account
```

```
*        number, ACCONT, using the value of ACCODE read.
*****************************************************************
   NUMELE = 21
   DO 65 I = 1,NUMELE
     READ(RUNIT1,70)NACONT(I)
     DO 60 J = 1,NACONT(I)
       READ(RUNIT1,100)ACCODE,PRCENT(I,J)
       ACCONT(I,J)=ACCNUM(ACCODE)
60   CONTINUE
65 CONTINUE
*****************************************************************
*
*        Read maintenance cost and operation cost information from link/node
*        data file.
*
*****************************************************************
   READ(RUNIT1,170)((MBCOST(I),MGRAD(I),MINST(I),MLIFE(I)),I=1,21)
*****************************************************************
*****************************************************************
   READ(RUNIT1,170)((OBCOST(I),OGRATE(I),OINST(I),OLIFE(I)),I=1,21)
*****************************************************************
*****************************************************************
   PRINT*
   PRINT*,'COMPLETED...'

   CLOSE (RUNIT1)
   RETURN

*****************************************************************
*        FORMAT STATEMENTS
*****************************************************************

40 FORMAT(F8.2)
42 FORMAT(6F8.2)
44 FORMAT(9F8.2)
45 FORMAT(F8.2)
70 FORMAT(I10)
80 FORMAT(F10.1,I10)
100        FORMAT(I10,F10.2)
170        FORMAT(3F10.2,I10)
175        FORMAT(A12,A35)

   END
```

B.2.19 SUBROUTINE: INDAT2

```
*                DESIGN DATA (This number will usually be a 5).
*
*   INTEREST DATA
*
*  NINT     - TOTAL # OF INTEREST RATE VALUES TO FOLLOW
*  INTRST   - VALUE OF THE INTEREST RATE (i.e. 12.50% , 9.75% )
*  LIFE     - EXPECTED LIFE OF THE PROJECT (YRS)
*
*   NODE DATA
*
*       NNODE    - TOTAL # OF NODES
*  NTYPE    - TYPE OF NODE, WHERE:
*             1 = TERMINATION
*             2 = DIRECTIONAL
*             3 = 3-WAY STREET INTERSECTION
*             4 = 4-WAY STREET INTERSECTION
*                  5 = OTHER
*  ELEV     - NATURAL GROUND ELEVATION OF NODE (M)
*
*   LINK DATA
*
*       NLINK    - TOTAL # OF LINKS
*  NODE1    - ORIGIN NODE OF LINK
*  NODE2    - DESTINATION NODE OF LINK
*  LLINK    - LENGTH OF LINK (M)
*  UNITS    - TOTAL # OF SINGLE FAMILY UNITS ON LINK
*  MULTI    - TOTAL # OF MULTI FAMILY UNITS ON LINK
*  AREA     - TOTAL DRAINAGE AREA OF LINK (HA)
*  LROAD    - LENGTH OF ROAD (M)
*  WROAD    - WIDTH OF ROAD (M)
*  TROAD    - TYPE OF ROAD, WHERE:
*             1 = COLLECTOR STREET
*             2 = LOCAL STREET
*  WWALK    - WIDTH OF SIDEWALK (M)
*  BARIER   - LENGTH OF BARRIER CURB & GUTTER ($/M)
*  ROLLED   - LENGTH OF ROLLED CURB & GUTTER ($/M)
*  BLVD     - LENGTH OF BLVD CURB & GUTTER ($/M)
*
*   TRIBUTARY DATA
*
*  NWM      - TOTAL # OF DESTINATION LINKS DESCRIBING THE
*             WATERMAIN SYSTEM
*  NDS      - TOTAL # OF DESTINATION LINKS DESCRIBING THE
*             DOMESTIC SEWER SYSTEM
*  NSS      - TOTAL # OF DESTINATION LINKS DESCRIBING THE
*             STORM SEWER SYSTEM
*  TBTRYW   - LOCATIONS (1 - 3) CONTAIN THE ORIGIN LINKS DESCRIBING
*             THE WATERMAIN TRIBUTARIES
*           - LOCATION (4) CONTAINS THE DESTINATION LINK
*  TBTRYD   - LOCATIONS (1 - 3) CONTAIN THE ORIGIN LINKS DESCRIBING
*             THE DOMESTIC SEWER TRIBUTARIES
*           - LOCATION (4) CONTAINS THE DESTINATION LINK
*  TBTRYS   - LOCATIONS (1 - 3) CONTAIN THE ORIGIN LINKS DESCRIBING
*             THE STORM SEWER TRIBUTARIES
*           - LOCATION (4) CONTAINS THE DESTINATION LINK
```

```
*  IQ       - TOTAL # OF LINKS ASSIGNED AN INITIAL VALUE, TO FOLLOW:
*  L        - THE NUMBER OF THE LINK
*  P        - INITIAL POPULATION OF LINK FOR WATERMAIN CALCULATION
*  PDS      - INITIAL POPULATION OF LINK FOR DOMESTIC SEWER
*             CALCULATION
*  ASS      - INITIAL DRAINAGE AREA OF LINK FOR STORM SEWER
*             CALCULATION (HECTARES)
*  AP       - TEMPORARY STORAGE FOR THE ABOVE VARIABLES.
*  APDS
*  AASS
*
*   APARK     - TOTAL AREA OF LAND DEDICATED TO PARK (HECTARE)
*   ABUFF     - TOTAL AREA OF LAND DEDICATED TO BUFFER ZONES (HECTARE)
*
*
*   LOCAL VARIABLES
*
*      ACCODE       - CODE CORRESPONDING TO AN ACCOUNT NAME AND NUMBER. (EG.
*             ACCODE=5 FOR ACCNUM(5) AND DEFINE(5).)
*      NUMELE       - NUMBER OF SUBDIVISION ELEMENTS.
*      I,J       - LOOP CONTROL VARIABLES
*      L,N       - LOOP CONTROL VARIABLES
*                  where: L refers to the link number.
*                         N refers to the node number.
*  D1-4     - Dummy variables to read end of data section flags (-1)
*  ID1-4
*
   SUBROUTINE INDAT2
*
   PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10
   INTEGER
     &  TROAD,TBTRYW,TBTRYD,TBTRYS,RUNIT2,RUNIT1,WUNIT1,WUNIT2,IST,
     &  WLINKS,NUMELE,ACCODE,OLIFE
   REAL
     &  LLINK,LROAD,LENGTH,MULTI,INTRST,LWALK,PIPIDX,IMCOST,MGRAD,MGRATE,
     &  MNCOST,MINST,MNPACC,IOCOST,MBCOST

   CHARACTER*12 ACCONT,ACCNUM
   CHARACTER*50,DEFINE

   COMMON
     &      /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &  UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA5/LWALK(LINNUM),WWALK(LINNUM),BARIER(LINNUM),ROLLED(LINNUM
     &      ,BLVD(LINNUM)
     &      /AREA6/NWMLNK,WLINKS(LINNUM),NWMGRP,TBTRYW(LINNUM,4)
     &  /AREA7/NDSLNK,DLINKS(LINNUM),NDSGRP,TBTRYD(LINNUM,4)
     &      /AREA8/NSSLNK,SLINKS(LINNUM),NSSGRP,TBTRYS(LINNUM,4)
     &      /AREA9/IQ,L,P(LINNUM),PDS(LINNUM),ASS(LINNUM)
     &      /AREA10/APARK,ABUFF
     &      /AREA17/COST25,COST26,COST27,COST28,COST29,COST30,COST31,
     &      COST32,COST33,MBCOST(25),OBCOST(25)
     &      /AREA20/ERRFLG
```

```
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &  /AREA36/LOPNUM,LOOPS(25,50,2),FLOW(150),FLODIR(150),
     &  HEDLOS(150),PIPDIA(150),HEADS(100),PSEHED(25),PIPIDX(10)
     &      /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
     &      ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
     &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),sdcost(25,10),
     &      MNPACC(25,10),OPPACC(25,10)
     &      /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
     &      ,DEFINE(100),NACCNT
     &      /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25),
     &      MNCOST(25,3),MCAPAF(25,10,3)
     &      /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25),
     &      OPCOST(25,3),OCAPAF(25,10,3)

      PRINT*
      PRINT*
      PRINT*,'L I N K / N O D E   D A T A '
      PRINT*
      IST=1
      CALL IOFILE (RUNIT2,IST)
      PRINT*
      PRINT*,'READING LINK/NODE DATA'

** * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Reads the node type, elevation, radius
*           values.
** * * * * * * * * * * * * * * * * * * * * * * * * * *
*
      N=0
      DO WHILE (NTYPE(N).NE.99)
              N=N+1
              READ(RUNIT2,100)NTYPE(N),ELEV(N),RADIUS(N)
      END DO
      IF (NTYPE(N).EQ.99)THEN
              N=N-1
              NNODE=N
      ENDIF
*
** * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Reads the link and area data
*           values.
** * * * * * * * * * * * * * * * * * * * * * * * * * *
*
      L=0
      DO WHILE (NODE1(L).NE.-1)
              L=L+1
      READ(RUNIT2,110)NODE1(L),NODE2(L),LLINK(L),UNITS(L),
        &  MULTI(L),AREA(L),FRONT(L)
      END DO
      IF (NODE1(L).EQ.-1)THEN
              L=L-1
              NLINK=L
      ENDIF
*
** * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Reads the road data
*           values.
```

```
** * * * * * * * * * * * * * * * * * * * * * * * *
*
   L=0
   READ(RUNIT2,120)((LROAD(L),WROAD(L),TROAD(L)),L=1,NLINK)
*
** * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Reads the sidewalk and curb
*           values.
** * * * * * * * * * * * * * * * * * * * * * * * * * * * *
   L=0
   READ(RUNIT2,130)((WWALK(L),BARIER(L),ROLLED(L),BLVD(L)),L=1,NLINK)
*
** * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Reads the population data
*           values.
** * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
   L=0
   DO WHILE (L.NE.-1)
   READ(RUNIT2,156)L,AP,APDS,AASS
   IF (L.NE.-1)THEN
           P(L)=AP
           PDS(L)=APDS
           ASS(L)=AASS
   ENDIF
   END DO
** * * * * * * * * * * * * * * * * * * * * * * * * * * *
*  Reads the 'other' subdivision area data
*           values.
** * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
   READ(RUNIT2,160)APARK
   READ(RUNIT2,160)ABUFF
   READ(RUNIT2,160)ASCHOL
   READ(RUNIT2,160)AMULT
   READ(RUNIT2,160)ACOM
   READ(RUNIT2,160)AOTHER
   READ(RUNIT2,160)ASUB
   READ(RUNIT2,166)NDUMMY
*  IF (NDUMMY.NE.-1)THEN
*    PRINT*
*    PRINT*,'FILE INCOMPLETE!!!'
*    PRINT*,'CHECK THE END OF THE DATAFILE FOR -1'
*    PRINT*,'THIS INDICATES COMPLETENESS.'
*  ENDIF
*
   PRINT*
   PRINT*,'COMPLETED...'
   CLOSE (RUNIT2)
70 FORMAT(I10)
80 FORMAT(F10.1,I10)
100        FORMAT(I10,2F10.2)
110        FORMAT(2I10,5F10.2)
120        FORMAT(2F10.2,I10)
125        FORMAT(2F10.2,I10)
130        FORMAT(4F10.1)
155        FORMAT(I10)
```

```
156         FORMAT(I10,3F10.2)
160         FORMAT(F10.2)
165         FORMAT(F10.2)
166         FORMAT(I10)
170         FORMAT(3F10.2,I10)
175         FORMAT(A12,A35)
   RETURN
   END
```

B.2.20 SUBROUTINE: IOFILE

```
*****************************************************************************

   SUBROUTINE IOFILE(DEVICE,IST)

   INTEGER DEVICE,IST
   CHARACTER*40 FNAME,FLAG*3

   IF ( IST.EQ.1 ) THEN

           PRINT*,'Please, enter the name of the INPUT file:'
           READ 55,FNAME
           FLAG = 'OLD'
55         FORMAT (A)

   ENDIF

   IF ( IST.EQ.2 ) THEN

           PRINT*,'Please, enter the name of the OUTPUT file:'
           READ 56,FNAME
           FLAG = 'NEW'
56         FORMAT (A)

   ENDIF

   OPEN ( UNIT=DEVICE, FILE=FNAME, STATUS=FLAG, ERR=60 )

   RETURN

60 WRITE (5,70)
70 FORMAT (1X,' F I L E   N O T   F O U N D ')

   STOP

   END
```

B.2.21 SUBROUTINE: LIGHT

```
      SUBROUTINE LIGHT


      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,ATTLIN,SS,CUL
        &     CNODE,CLINK,PATH,BRANCH,STRLIT,LINASS(4),LINK(3)

      REAL
        &  LLINK,LROAD,LENGTH,KM,LTCOSL,LTCOSN

      COMMON
        &     /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
        &     ANGLE(NODNUM)
        &     /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
        &     UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
        &     /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
        &     /AREA11/COST1,COST2/AREA12/COST3,COST4/AREA13/COST5,COST6
        &     /AREA19/TLCOST
        &     /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &     /AREA25/SPAMAN,SPALIT,SPACAT
        &     /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
        &     /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
        &     /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
        &     /AREA34/STRLIT(NODNUM,NODNUM,4),LTCOSL(LINNUM),LTCOSN(NODNUM)
        &     LTLINK(LINNUM),LTNODE(NODNUM)
        &     /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,SLCOST,TSIGN,CSIGN,MNHOLE,
        &     MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
        &     WKCOST,BRCOST,RLCOST,BLCOST
*
*
*
      DO 50 N1=1,NNODE
*     set the sub-total ( per node ) to zero
            LTCOSN(N1)=0.0
            LTNODE(N1)=0
            DO 60 N2=1,NNODE
              DO 70 J=1,4
*     set the number of street light per street length ( city block ) to zero
                STRLIT(N1,N2,J)=0
70            CONTINUE
60          CONTINUE
50 CONTINUE
*
      DO 80 L=1,NLINK
*     set the sub-total ( per link ) to zero
            LTCOSL(L)=0.0
            LTLINK(L)=0
80 CONTINUE
*
      CALL COUNTER
*
*
```

```
*  check the type of the node
*  if the node type is intersection or dead end/cul-de-sac
*  then find the links in between the nodes
   DO 100 N=1,NNODE
           IF ( NTYPE(N).EQ.3 .OR. NTYPE(N).EQ.4 )  THEN
*
*  there is four possible branches for each node
             DO 110 J=1,4
*  set the number of links flag to zero
               NL=0
               CALL FINDPATH(N,J,NL)
               IF ( NL.EQ.0 )  GOTO 110
               CLINK=PATH(N,J,NL)
*
*
*  set up the subscript for the variable STRLEN to store
*  the street length between intersection/intersection, or
*  intersection/dead end (cul-de-sac)
               IF ( NTYPE(NODE1(CLINK)).NE.2 .AND. NTYPE(NODE2(CLINK))
     &                  .NE.2 )  THEN
                 IF ( N.EQ.NODE1(CLINK) )  THEN
                   N2=NODE2(CLINK)
                 ELSE
                   N2=NODE1(CLINK)
                 ENDIF
               ELSE
                 IF ( NTYPE(NODE1(CLINK)).EQ.2 )  THEN
                   N2=NODE2(CLINK)
                 ELSE
                   N2=NODE1(CLINK)
                 ENDIF
               ENDIF
               N1=N
*
*  set the total street length between intersection/intersection, or
*  intersection/dead end (cul-de-sac) to zero
               TOTAL=0.0
*
*  find the street length ( or city block ) by chasing all the links
*  in between the nodes N1,N2
               DO 130 JJ=1,NL
                 CLINK=PATH(N,J,JJ)
*  find correct street length
                 CALL CHECKLEN(CLINK,SLEN)
                 TOTAL=TOTAL+SLEN
130                    CONTINUE
*
*
               STRLEN(N1,N2,J)=TOTAL
110                  CONTINUE
           ENDIF
100        CONTINUE
*
*
*
*  put the subscripts in increasing order
   DO 140 N1=1,NNODE
```

```
            DO 143 N2=1,NNODE
              DO 145 J=1,4
                IF ( STRLEN(N1,N2,J).NE.0.0 )  THEN
*  find the number of links betwwen the given nodes
                  CALL FINDPATH(N1,J,NL)
                  CLINK=PATH(N1,J,NL)
                  DO 147 JJ=1,4
                    IF ( ATTLIN(N2,JJ).EQ.CLINK )  THEN
                      J2=JJ
                    ENDIF
147                     CONTINUE
                  IF ( N1.LT.N2 .AND. STRLEN(N2,N1,J2).EQ.0.0 )  THEN
                    STRLEN(N2,N1,J2)=STRLEN(N1,N2,J)
                  ENDIF
                ENDIF
145                 CONTINUE
143               CONTINUE
140         CONTINUE
*
*
*
*  calculate number of street lights required in the sub-division
*
*  set the regular street light counter to zero
   NRGLIT=0
*  set the cul-de-sac street light counter to zero
   NSALIT=0
*
*
*  add enough street light to the intersection
*  and cul-de-sac
   DO 150 N1=1,NNODE
*
*  street lights for cul-de-sac
*  CHANGE THE IF STATEMENT WHEN THE DATA FILE IS SET UP CORRECTLY
*  '.AND.'
            IF ( RADIUS(N1).NE.0.0 .OR. NTYPE(N1).EQ.1 )  THEN
              NSALIT=NSALIT+1
              LTNODE(N1)=1
              LTCOSN(N1)=COST2
              DO 155 J=1,4
                CLINK=ATTLIN(N1,J)
                IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                  SS=J
                ENDIF
155                 CONTINUE
              CLINK=ATTLIN(N1,SS)
              IF ( NTYPE(NODE1(CLINK)).EQ.2 .OR. NTYPE(NODE2(CLINK))
     &                .EQ.2 )  THEN
                CALL CULCHK(CLINK,CUL,M1,M2,LINK1,A)
                IF ( CUL.EQ.0 )  THEN
                  LTCOSL(CLINK)=COST2+LTCOSL(CLINK)
                  LTLINK(CLINK)=LTLINK(CLINK)+1
                ELSE
                  LTCOSL(LINK1)=COST2+LTCOSL(LINK1)
                  LTLINK(LINK1)=LTLINK(LINK1)+1
                ENDIF
```

```
              ELSE
                LTCOSL(CLINK)=COST2+LTCOSL(CLINK)
                LTLINK(CLINK)=LTLINK(CLINK)+1
              ENDIF
            ENDIF
*  street lights for 3-4 way intersection
            IF ( NTYPE(N1).EQ.3 .OR. NTYPE(N1).EQ.4 )  THEN
              NRGLIT=NRGLIT+( NTYPE(N1)-2 )
              LTNODE(N1)=NTYPE(N1)-2
              LTCOSN(N1)=LTNODE(N1)*COST2
              CALL DMAKER (N1,LTNODE(N1),LINK)
              DO 156 J=1,LTNODE(N1)
                LTCOSL(LINK(J))=COST2+LTCOSL(LINK(J))
                LTLINK(LINK(J))=LTLINK(LINK(J))+1
156                   CONTINUE
            ENDIF
*
*  intersection to intersection
            DO 160 N2=1,N1
              IF ( N1.EQ.N2 )  THEN
                BRANCH=1
              ELSE
                BRANCH=4
              ENDIF
              DO 170 J=1,4
                IF ( STRLEN(N1,N2,J).NE.0.0 )  THEN
                  IF ( BRANCH.GT.0 )  THEN
                    BRANCH=BRANCH-1
*  set the number of links between two intersections or intersection/dead
*  end/cul-de-sac to zero
                    NBL=0
*  determine the number of links between two intersections or
*  intersection/dead end/cul-de-sac
175                         NBL=NBL+1
                    CLINK=PATH(N1,J,NBL)
                    IF ( CLINK.EQ.0 )  THEN
                      NBL=NBL-1
                    ELSE
                      GOTO 175
                    ENDIF
*  determine the correct distance for placing street lights
                    CALL THREEWAY(N1,N2,J,NBL,EXRLEN)
*
*  determine the number of street lights per street length
*  ( or city block )
                    LIGHTS=INT((STRLEN(N1,N2,J)+EXRLEN)/SPALIT-0.5)
                    NRGLIT=LIGHTS+NRGLIT
                    STRLIT(N1,N2,J)=LIGHTS
*  determine which link has the street lights
*  the total length by adding the spacing of street lights
                    STLEN1=0.0
*  the total length by adding the road length of the link
                    STLEN2=EXRLEN
*
*  set up the start counter
                    K=0
*
```

```
                    STLEN1=STLEN1+SPALIT
180                         IF ( STLEN1.LE.STLEN2 )  THEN
                      CLINK=PATH(N1,J,K)
                      IF ( LIGHTS.GT.0 )  THEN
                        LTLINK(CLINK)=LTLINK(CLINK)+1
                        LIGHTS=LIGHTS-1
                        LTCOSL(CLINK)=LTCOSL(CLINK)+COST2
                        STLEN1=STLEN1+SPALIT
                        GOTO 180
                      ENDIF
                    ELSE
                      K=K+1
                      IF ( K.LE.NBL)  THEN
                        CLINK=PATH(N1,J,K)
                        CALL CHECKLEN(CLINK,SLEN)
                        STLEN2=STLEN2+SLEN
                        GOTO 180
                      ENDIF
                    ENDIF
                  ENDIF
                ENDIF
170                   CONTINUE
160                 CONTINUE
150         CONTINUE
*
*
*  add up all different type of street lights
   NLIGHT=NRGLIT+NSALIT
*
*  the price for the cul-de-sac street lights are unknown????????
*  therefore, assume all street lights have the same price
*
   SLCOST=NLIGHT*COST2
*
   CALL ASPLIT(13,SLCOST)
*
   TLCOST=TLCOST+SLCOST
*
*  WRITE(WUNIT1,640)NLIGHT,SLCOST
640         FORMAT(10X,'STREET LIGHTS (#)',T40,I6,T54,F7.0)


*  OUTPUT THE RESULT (SPECIAL)

*
   T=0.0
   NLITS=0
   DO 152 L=1,NLINK
           T=T+LTCOSL(L)
           NLITS=NLITS+LTLINK(L)
152        CONTINUE
   WRITE(088,154)SLCOST,NLIGHT,T,NLITS
154        FORMAT ('1',' LIGHT :',2(3X,F9.0,3X,I3))

   DO 911 N1=1,NNODE
   DO 902 N2=1,N1
           DO 903 J=1,4
```

```
          IF ( STRLEN(N1,N2,J).NE.0.0 )  THEN
          WRITE (088,950) N1,N2,J,STRLEN(N1,N2,J),STRLIT(N1,N2,J)
          ENDIF
903                 CONTINUE
902       CONTINUE
911       CONTINUE
   DO 904 I=1,NNODE
   WRITE (088,900)I,LTNODE(I),LTCOSN(I)
904       CONTINUE
   DO 905 I=1,NLINK
   WRITE (088,931)I,LTLINK(I),LTCOSL(I)
905       CONTINUE

931       FORMAT (1X,'LINK',2(3X,I3),F9.3)
900       FORMAT (1X,'NODE',2(3X,I3),F9.3)
950       FORMAT (' ','FROM NODE ',3X,I3,3X,' TO NODE ',3X,I3,3X,' BRANCH
     &,3X,I3,3X,' LENGTH ',3X,F9.3,3X,'LIGHT ',3X,I3)

   RETURN
   END
```

B.2.22 SUBROUTINE: MAINCE

```
            SUBROUTINE MAINCE

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER TYPE,METHOD,ELEM,WUNIT1,TSIGN,TIME
        &       ,ICDFLG,IDAFLG,TRIFLG,CCTFLG,OPRFLG,MNTFLG,ECNFLG,ACCFLG

      REAL IMCOST,MGRAD,MGRATE,MNCOST,MINST,MNPACC,MCAPAF
        &       ,MBCOST

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
        &       /AREA10/APARK,ABUFF,PCOST,BCOST
        &       /AREA17/COST25,COST26,COST27,COST28,COST29,COST30,COST31,
        &       COST32,COST33,MBCOST(25),OBCOST(25)
        &       /AREA20/ERRFLG,ICDFLG,IDAFLG,TRIFLG,CCTFLG,OPRFLG,MNTFLG,ECNFLG,
        &       /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &       /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
        &       ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
        &       /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
        &       MHCOST,NBASIN,CBCOST,NTRAFF,TFCOST,TWK,TBR,TRL,TBL,
        &       WKCOST,BRCOST,RLCOST,BLCOST
        &       /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &       ,MNPACC(25,10),OPPACC(25,10)
        &       /AREA47/NACCNT,ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &       ,DEFINE(100)
        &       /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &       ,MNCOST(25,3),MCAPAF(25,10,3)
        &       /AREA51/TPAREA,SSPIPE,WMPIPE,DSPIPE
*
      TIME=3
*
*
      DO 100 TYPE=1,21
*  determine which method is going to be used
            TPCENT=0.0
            DO 110 I=1,NACONT(TYPE)
                    TPCENT=TPCENT+PRCENT(TYPE,I)
110                 CONTINUE
            IF ( TPCENT.LE.1 ) THEN
*  using the ratio or percentage
              METHOD=1
            ELSE
*  the given cost are already broke down
              METHOD=2
            ENDIF
*
*  find the present,annual,future worth
            IF ( MGRAD(TYPE).EQ.0.0 )  THEN
              CALL GOGRAD (IMCOST(TYPE),MGRATE(TYPE),MINST(TYPE),MLIFE(TY
        &                  ,P,A,F)
              MNCOST(TYPE,1)=P
```

```
              MNCOST(TYPE,2)=A
              MNCOST(TYPE,3)=F
            ELSE
              MNCOST(TYPE,1)=ARITHP(IMCOST(TYPE),MGRAD(TYPE),MINST(TYPE)
     &                ,MLIFE(TYPE))
              MNCOST(TYPE,2)=ARITHA(IMCOST(TYPE),MGRAD(TYPE),MINST(TYPE)
     &                ,MLIFE(TYPE))
              MNCOST(TYPE,3)=ARITHF(IMCOST(TYPE),MGRAD(TYPE),MINST(TYPE)
     &                ,MLIFE(TYPE))
            ENDIF

*  apply the correct method
            IF ( METHOD.EQ.1 )  THEN
              DO 120 I=1,NACONT(TYPE)
                MNPACC(TYPE,I)=IMCOST(TYPE)*PRCENT(TYPE,I)
                MCAPAF(TYPE,I,1)=MNCOST(TYPE,1)*PRCENT(TYPE,I)
                MCAPAF(TYPE,I,2)=MNCOST(TYPE,2)*PRCENT(TYPE,I)
                MCAPAF(TYPE,I,3)=MNCOST(TYPE,3)*PRCENT(TYPE,I)
                DO 130 J=1,NACCNT
                  IF ( ACCNUM(J).EQ.ACCONT(TYPE,I) )  THEN
                    ELEM=J
                  ENDIF
130                     CONTINUE
                AMCOST(ELEM)=AMCOST(ELEM)+MNPACC(TYPE,I)
120                   CONTINUE
            ELSE
              DO 140 I=1,NACONT(TYPE)
                DO 150 J=1,NACCNT
                  IF ( ACCNUM(J).EQ.ACCONT(TYPE,I) )  THEN
                    ELEM=J
                  ENDIF
150                     CONTINUE
                AMCOST(ELEM)=AMCOST(ELEM)+PRCENT(TYPE,I)
140                   CONTINUE
            ENDIF
100         CONTINUE
*
*  find the total main. cost
   TMCOST=0.0
   DO 160 I=1,25
            TMCOST=IMCOST(I)+TMCOST
160         CONTINUE

*  summary
*
*
   IF ( MNTFLG.EQ.0 )  THEN
   WRITE (WUNIT1,900)
   DO 257 I=1,TIME
            WRITE (WUNIT1,905)
257         CONTINUE
   WRITE (WUNIT1,910)
   DO 258 I=1,TIME
            WRITE (WUNIT1,911)
258         CONTINUE
   WRITE (WUNIT1,915)
   DO 259 I=1,TIME
```

```
            WRITE (WUNIT1,920)
259         CONTINUE
*
   WRITE (WUNIT1,925) WMPIPE,MBCOST(1),IMCOST(1)
   WRITE (WUNIT1,930) DSPIPE,MBCOST(2),IMCOST(2)
   WRITE (WUNIT1,935) SSPIPE,MBCOST(3),IMCOST(3)
   WRITE (WUNIT1,940) TPAREA,MBCOST(4),IMCOST(4)
   WRITE (WUNIT1,955) TWK,MBCOST(5),IMCOST(5)
   WRITE (WUNIT1,960) TBR,MBCOST(6),IMCOST(6)
   WRITE (WUNIT1,965) TRL,MBCOST(7),IMCOST(7)
   WRITE (WUNIT1,970) TBL,MBCOST(8),IMCOST(8)
   WRITE (WUNIT1,1063) APARK,MBCOST(9),IMCOST(9)
   WRITE (WUNIT1,1065) ABUFF,MBCOST(10),IMCOST(10)
   WRITE (WUNIT1,985) NBASIN,MBCOST(11),IMCOST(11)
   WRITE (WUNIT1,990) MNHOLE,MBCOST(12),IMCOST(12)
   WRITE (WUNIT1,1010) NLIGHT,MBCOST(13),IMCOST(13)
   WRITE (WUNIT1,1015) TSIGN,MBCOST(14),IMCOST(14)
   WRITE (WUNIT1,1020) NTRAFF,MBCOST(15),IMCOST(15)
   WRITE (WUNIT1,1030) INT(TSING)
   WRITE (WUNIT1,1035) MBCOST(16),IMCOST(16)
   WRITE (WUNIT1,1040) MBCOST(17),IMCOST(17)
   WRITE (WUNIT1,1045) MBCOST(18),IMCOST(18)
   WRITE (WUNIT1,1050) INT(TMULT)
   WRITE (WUNIT1,1035) MBCOST(19),IMCOST(19)
   WRITE (WUNIT1,1040) MBCOST(20),IMCOST(20)
   WRITE (WUNIT1,1045) MBCOST(21),IMCOST(21)
   WRITE (WUNIT1,1070) TMCOST
   DO 260 I=1,TIME
            WRITE (WUNIT1,1080) TMCOST
260         CONTINUE
   ENDIF
*  30,20,20,25
900         FORMAT ('1',' M A I N T E N A N C E   C O S T')
905         FORMAT ('+',' M A I N T E N A N C E   C O S T')
910         FORMAT (1X,' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
911         FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
915         FORMAT (/////,6X,'DESCRIPTION',20X,'QUANTITY',17X,'UNIT COST',
     &      9X,'MAINTENANCE COST')
920         FORMAT ('+',5X,'DESCRIPTION',20X,'QUANTITY',17X,'UNIT COST',
     &      9X,'MAINTENANCE COST')
925         FORMAT (//,6X,'WATERMAIN',19X,F10.2,'m',6X,2(8X,'$',1X,F10.2,5X)
930         FORMAT (6X,'DOMESTIC SEWER',14X,F10.2,'m',6X,2(8X,'$',1X,F10.2,5
935         FORMAT (6X,'STORM SEWER',17X,F10.2,'m',6X,2(8X,'$',1X,F10.2,5X))
940         FORMAT (//,6X,'PAVEMENT',20X,F10.2,'sq. m',2X,2(8X,'$',1X,F10.2,
955         FORMAT (//,6X,'SIDEWALK',20X,F10.2,'sq. m',2X,2(8X,'$',1X,F10.2,
960         FORMAT (6X,'BARRIER CURB & GUTTER',7X,F10.2,'m',
     &      6X,2(8X,'$',1X,F10.2,5X))
965         FORMAT (6X,'ROLLED CURB & GUTTER',8X,F10.2,'m',
     &      6X,2(8X,'$',1X,F10.2,5X))
970         FORMAT (6X,'BLVD. CURB & GUTTER',9X,F10.2,'m',
     &      6X,2(8X,'$',1X,F10.2,5X))
985         FORMAT (//,6X,'CATCH-BASIN',20X,I4,2X,' units',
     &      2X,2(8X,'$',1X,F10.2,5X))
990         FORMAT (//,6X,'MANHOLE',24X,I4,2X,' units',
     &      2X,2(8X,'$',1X,F10.2,5X))
1010        FORMAT (//,6X,'STREET LIGHT',19X,I4,2X,' units',
     &      2X,2(8X,'$',1X,F10.2,5X))
```

```
1015        FORMAT (6X,'STREET SIGN',20X,I4,2X,' units',
     &      2X,2(8X,'$',1X,F10.2,5X))
1020        FORMAT (6X,'TRAFFIC SIGN',19X,I4,2X,' units',
     &      2X,2(8X,'$',1X,F10.2,5X))
1030        FORMAT (//,6X,'SINGLE FAMILY',18X,I4,2X,' units')
1035        FORMAT (10X,'- GAS',36X,2(8X,'$',1X,F10.2,5X))
1040        FORMAT (10X,'- POWER',34X,2(8X,'$',1X,F10.2,5X))
1045        FORMAT (10X,'- TELEPHONE',30X,2(8X,'$',1X,F10.2,5X))
1050        FORMAT (6X,'MULTI-FAMILY',19X,I4,2X,' units')
1063        FORMAT (//,6X,'PARK DEVELOPMENT',12X,F10.2,'Ha',
     &      5X,2(8X,'$',1X,F10.2,5X))
1065        FORMAT (6X,'BUFFER DEVELOPMENT',10X,F10.2,'Ha',
     &      5X,2(8X,'$',1X,F10.2,5X))
1070        FORMAT (//,11X,'T O T A L',64X,'$',1X,F10.2)
1080        FORMAT ('+',10X,'T O T A L',64X,'$',1X,F10.2)
*
*
   RETURN
   END
```

B.2.23 SUBROUTINE: MANHOLE

```
      SUBROUTINE MANHOLE


*****************************************************************
*                                                               *
*                                                               *
*                            MANHOLE                            *
*                          REQUIREMENTS                         *
*                                                               *
*                                                               *
*****************************************************************
*
*
*
*  NTYPE  - node type.
*  ATTLIN - the links associate with the given node. ( NOTE: max. links can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  WROAD  - width of the road of a given link.
*  HOLE   - number of manholes require per link per sewer.
*  DSMAN  - the number of domestic manholes per link.
*  DSCOSL - the cost of domestic manholes per link.
*  DSCOSN - the cost of domestic manhole per node.
*  SSMAN  - the number of storm manholes per link.
*  SSCOSL - the cost of storm manholes per link.
*  SSCOSN - the cost of storm manhole per node.
*  RADIUS - the radius of the cul-de-sac/bulb curve/tube sac
*  MNHOLE - the total of number of manholes require in the sub-division.
*  MHCOST - the total cost of the manholes for the sub-division.
*  DSLINK - domestic sewer elev. of a given link.
*  SSLINK - storm sewer elev. of a given link.
*  NODE1  - origin node of the current link.
*  N1     - origin node of the current link.
*  NODE2  - dest. node of the current link.
*  N2     - dest. node of the current link.
*  SSELE1 - storm sewer elev. of the current link at the origin node.
*  SSELE2 - storm sewer elev. of the current link at the dest. node.
*  DSELE1 - domestic sewer elev. of the current link at the origin node.
*  DSELE2 - domestic sewer elev. of the current link at the dest. node.
*  DPTH  - depth of the manhole from the ground level.
*  ELEV1  - the elevation difference at origin node.
*  ELEV2  - the elevation difference at dest node.
*  SPAMAN - spacing of catch-basin
*  MANBED - the height of the bedding.
*  LENGTH - length of the link
*  CUL    - cul-de-sac type 2 node flag
*  I,J    - loop counter
*  SS     - subscript
*
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10
```

```
      INTEGER
     &  ATTLIN,SS,RUNIT1,RUNIT2,WUNIT1,WUNIT2,DSMAN,SSMAN,HOLE
     &      ,CLINK,CUL,LINK(2)

      REAL
     &  LLINK,LENGTH,MHCOST,MANBED

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
     &      /AREA19/TLCOST
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA23/DSCOVR,DSVMIN,DSVMAX
     &      /AREA24/SSCOVR,SSVMIN,SSVMAX
     &      /AREA25/SPAMAN,SPALIT,SPACAT
     &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
     &      /AREA29/DSMAN(LINNUM),SSMAN(LINNUM),DSCOSL(LINNUM),
     &      SSCOSL(LINNUM),DSCOSN(NODNUM),SSCOSN(NODNUM)
     &      /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
     &      /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
     &      MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
     &      WKCOST,BRCOST,RLCOST,BLCOST
*
*
*
*  temporary value of the bedding
      MANBED=.200
*
*
*
*  set the total of manholes and the cost to zero
      MNHOLE=0
      MHCOST=0.0
*
*  set the cost of an manhole per node to zero
      DO 90 I=1,NNODE
              DSCOSN(I)=0.0
              SSCOSN(I)=0.0
90    CONTINUE
*
*  set the cost and number of manholes to zero for every link
      DO 100 I=1,NLINK
              DSCOSL(I)=0.0
              DSMAN(I)=0
              SSCOSL(I)=0.0
              SSMAN(I)=0
100           CONTINUE
```

```
*
*
*
      DO 110 I=1,NLINK
*
*     set the cul-de-sac flag to zero
      CUL=0
      IF ( NTYPE(NODE1(I)).EQ.2 .OR. NTYPE(NODE2(I)).EQ.2 )  THEN
              CALL CULCHK(I,CUL,N1,N2,LINK1,LENGTH)
      ENDIF
*
      IF ( CUL.EQ.1 )  THEN
              CLINK=LINK1
      ELSE
              CLINK=I
              N1=NODE1(I)
              N2=NODE2(I)
              LENGTH=LLINK(I)
      ENDIF
*
*     calculate number of manholes require per link
              HOLE=INT(LENGTH/SPAMAN-0.5)
*     since the link is too short, therefore no manhole requires
              IF ( HOLE.LT.1 )  GOTO 110
*
*     determine the depth and the cost of the manholes
*
*     for DOMESTIC MANHOLE
*     determine the slope angle of the pipe.
*
*     find the domestic sewer elev. at origin node
              DO 120 J=1,4
                IF ( ATTLIN(N1,J).EQ.I )  THEN
                  SS=J
                ENDIF
120                   CONTINUE
              DSELE1=DSLINK(N1,SS)
*
*     find the domestic sewer elev. at dest. node
              DO 130 J=1,4
                IF ( ATTLIN(N2,J).EQ.I )  THEN
                  SS=J
                ENDIF
130                   CONTINUE
              DSELE2=DSLINK(N2,SS)
              IF ( DSELE1.LT.9000.0 .AND. DSELE2.LT.9000.0 )  THEN
                MNHOLE=MNHOLE+HOLE
                DSMAN(CLINK)=HOLE
*     determine the domestice pipe slope angle
                ELEV1=ELEV(N1)-DSELE1
                ELEV2=ELEV(N2)-DSELE2
                IF ( ELEV1.LT.ELEV2 )  THEN
                  ANG=ATAN((ELEV2-ELEV1)/LENGTH)
*
*     determine the price for each manhole per link
                  DO 140 M=1,HOLE
                    DPTH=TAN(ANG)*M*SPAMAN+ELEV1+MANBED
```

```
                DSCOSL(CLINK)=DSCOSL(CLINK)+(COST15*DPTH)
140                   CONTINUE
            ELSE
              ANG=ATAN((ELEV1-ELEV2)/LENGTH)
*
*  determine the price for each manhole per link
              DO 150 M=1,HOLE
                DPTH=TAN(ANG)*(LENGTH-M*SPAMAN)+ELEV2+MANBED
                DSCOSL(CLINK)=DSCOSL(CLINK)+(COST15*DPTH)
150                   CONTINUE
            ENDIF
*  addup the total cost per link and the whole sub-division
            DSCOSL(CLINK)=DSCOSL(CLINK)+COST14
            MHCOST=MHCOST+DSCOSL(CLINK)
          ENDIF
*
*
*  for STORM MANHOLE
*  determine the slope angle of the pipe.
*
*  find the storm sewer elev. at origin node
          DO 160 J=1,4
            IF ( ATTLIN(N1,J).EQ.I )  THEN
              SS=J
            ENDIF
160                 CONTINUE
          SSELE1=SSLINK(N1,SS)
*
*  find the storm sewer elev. at dest. node
          DO 170 J=1,4
            IF ( ATTLIN(N2,J).EQ.I )  THEN
              SS=J
            ENDIF
170                 CONTINUE
          SSELE2=SSLINK(N2,SS)
          IF ( SSELE1.LT.9000.0 .AND. SSELE2.LT.9000.0 )  THEN
            MNHOLE=MNHOLE+HOLE
            SSMAN(CLINK)=HOLE
*  determine the storm pipe slope angle
            ELEV1=ELEV(N1)-SSELE1
            ELEV2=ELEV(N2)-SSELE2
            IF ( ELEV1.LT.ELEV2 )  THEN
              ANG=ATAN((ELEV2-ELEV1)/LENGTH)
*
*  determine the price for each manhole per link
              DO 180 M=1,HOLE
                DPTH=TAN(ANG)*M*SPAMAN+ELEV1+MANBED
                SSCOSL(CLINK)=SSCOSL(CLINK)+(COST15*DPTH)
180                   CONTINUE
            ELSE
              ANG=ATAN((ELEV1-ELEV2)/LENGTH)
*
*  determine the price for each manhole per link
              DO 190 M=1,HOLE
                DPTH=TAN(ANG)*(LENGTH-M*SPAMAN)+ELEV2+MANBED
                SSCOSL(CLINK)=SSCOSL(CLINK)+(COST15*DPTH)
190                   CONTINUE
```

```
              ENDIF
*  addup the total cost per link and the whole sub-division
              SSCOSL(CLINK)=SSCOSL(CLINK)+COST14
              MHCOST=MHCOST+SSCOSL(CLINK)
            ENDIF
110         CONTINUE
*
*
*
   DO 200 I=1,NNODE
*  add one manhole to the node and calculate the cost
*  check the type of the node, if the node is cul-de-sac type 2 node,
*  then skips the calculation
*  set the cul-de-sac flag to zero
            CUL=0
            IF ( NTYPE(I).EQ.2 )  THEN
              DO 205 J=1,4
                CLINK=ATTLIN(I,J)
                N1=NODE1(CLINK)
                N2=NODE2(CLINK)
                IF ( NTYPE(N1).EQ.1 .AND. RADIUS(N1).NE.0.0 .OR.
     &                    NTYPE(N2).EQ.1 .AND. RADIUS(N2).NE.0.0 )  THEN
                  CUL=1
                ENDIF
205                   CONTINUE
            ENDIF
*
            IF ( CUL.EQ.1 )  GOTO 200
*
*  for  DOMESTIC MANHOLE
*
*  find the lowest domestic elev. pipe line
            DSELE1=9999.0
            DO 210 J=1,4
              IF ( DSLINK(I,J).LT.DSELE1 .AND. DSLINK(I,J).NE.0.0 )  THEN
                DSELE1=DSLINK(I,J)
              ENDIF
210                 CONTINUE
*
            IF ( DSELE1.LT.9000.0 )  THEN
              MNHOLE=MNHOLE+1
              DPTH=ELEV(I)-DSELE1+MANBED
              DSCOSN(I)=COST14+COST15*DPTH
              MHCOST=MHCOST+DSCOSN(I)
              IF ( NTYPE(I).EQ.7 )  THEN
                DO 215 J=1,4
                  CLINK=ATTLIN(I,J)
                  IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                    SS=J
                  ENDIF
215                     CONTINUE
                CALL CULCHK(CLINK,CUL,M1,M2,LINK1,A)
                IF ( CUL.EQ.0 )  THEN
                  CALL DMAKER (I,1,LINK)
                  DSCOSL(LINK(1))=DSCOSN(I)+DSCOSL(LINK(1))
                  DSMAN(LINK(1))=DSMAN(LINK(1))+1
                ELSE
```

```
                DSCOSL(LINK1)=DSCOSN(I)+DSCOSL(LINK1)
                DSMAN(LINK1)=DSMAN(LINK1)+1
              ENDIF
            ELSE
              CALL DMAKER (I,1,LINK)
              DSCOSL(LINK(1))=DSCOSN(I)+DSCOSL(LINK(1))
              DSMAN(LINK(1))=DSMAN(LINK(1))+1
            ENDIF
          ENDIF
*
*
*  for STORM MANHOLE
*
*  find the lowest storm elev. pipe line
          SSELE1=9999.0
          DO 220 J=1,4
            IF ( SSLINK(I,J).LT.SSELE1 .AND. SSLINK(I,J).NE.0.0 )  THEN
              SSELE1=SSLINK(I,J)
            ENDIF
220               CONTINUE
*
          IF ( SSELE1.LT.9000.0 )  THEN
            MNHOLE=MNHOLE+1
            DPTH=ELEV(I)-SSELE1+MANBED
            SSCOSN(I)=COST14+COST15*DPTH
            MHCOST=MHCOST+SSCOSN(I)
            IF ( NTYPE(I).EQ.7 )  THEN
              DO 230 J=1,4
                CLINK=ATTLIN(I,J)
                IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                  SS=J
                ENDIF
230                   CONTINUE
              CALL CULCHK(CLINK,CUL,M1,M2,LINK1,A)
              IF ( CUL.EQ.0 )  THEN
                CALL DMAKER (I,1,LINK)
                SSCOSL(LINK(1))=SSCOSN(I)+SSCOSL(LINK(1))
                SSMAN(LINK(1))=SSMAN(LINK(1))+1
              ELSE
                SSCOSL(LINK1)=SSCOSN(I)+SSCOSL(LINK1)
                SSMAN(LINK1)=SSMAN(LINK1)+1
              ENDIF
            ELSE
              CALL DMAKER (I,1,LINK)
              SSCOSL(LINK(1))=SSCOSN(I)+SSCOSL(LINK(1))
              SSMAN(LINK(1))=SSMAN(LINK(1))+1
            ENDIF
          ENDIF
200       CONTINUE
*
   CALL ASPLIT(12,MHCOST)
*
   TLCOST=TLCOST+MHCOST
*  WRITE(WUNIT1,320)MNHOLE,MHCOST
320       FORMAT('1',9X,'MANHOLE (#)',T40,I6,T54,F7.0,/)
*
```

```
*  OUTPUT THE RESULT (SPECIAL)
   T=0.0
   NM=0

   DO 333 I=1,NLINK
   T=DSCOSL(I)+SSCOSL(I)+T
   NM=DSMAN(I)+SSMAN(I)+NM
333         CONTINUE

            WRITE (088,620)MNHOLE,NM
            WRITE (088,610)MHCOST,T
   DO 401 I=1,NNODE
            WRITE (088,601),I,DSCOSN(I),SSCOSN(I)
401         CONTINUE

   DO 400 I=1,NLINK
            WRITE (088,600)I,DSMAN(I),DSCOSL(I),
     &SSMAN(I),SSCOSL(I)

400         CONTINUE


620         FORMAT ('1',' MANHOLE =',2(3X,I3))
610         FORMAT (//,' MANHOLE COST =',3X,2(F10.0))
601         FORMAT (1X,'NODE :',I3,3X,'DS :',F10.3,3X,'SS :',F10.3)
600         FORMAT (1X,'LINK :',I3,3X,'DS : ',I3,3X,F10.3,3X,'SS : '
     &      ,I3,3X,F10.3)


   RETURN
   END
```

B.2.24 SUBROUTINE: OMINIT

```
      SUBROUTINE OMINIT

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER TSIGN

      REAL MBCOST,IMCOST,IOCOST


      COMMON
        &      /AREA10/APARK,ABUFF,PCOST,BCOST
        &      /AREA17/COST25,COST26,COST27,COST28,COST29,COST30,COST31,
        &      COST32,COST33,MBCOST(25),OBCOST(25)
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
        &      ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
        &      /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
        &      MHCOST,NBASIN,CBCOST,NTRAFF,TFCOST,TWK,TBR,TRL,TBL,
        &      WKCOST,BRCOST,RLCOST,BLCOST
        &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &      ,MNPACC(25,10),OPPACC(25,10)
        &      /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &      ,DEFINE(100)
        &      /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &      ,MNCOST(25,3),MCAPAF(25,10,3)
        &      /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &      ,OPCOST(25,3),OCAPAF(25,10,3)
        &      /AREA51/TPAREA,SSPIPE,WMPIPE,DSPIPE
*
*
*
*     main. cost
      IMCOST(1)=WMPIPE*MBCOST(1)
      IMCOST(2)=DSPIPE*MBCOST(2)
      IMCOST(3)=SSPIPE*MBCOST(3)
*
      IMCOST(4)=TPAREA*MBCOST(4)
      IMCOST(5)=TWK*MBCOST(5)
      IMCOST(6)=TBR*MBCOST(6)
      IMCOST(7)=TRL*MBCOST(7)
      IMCOST(8)=TBL*MBCOST(8)
*
      IMCOST(9)=APARK*MBCOST(9)
      IMCOST(10)=ABUFF*MBCOST(10)
*
      IMCOST(11)=NBASIN*MBCOST(11)
      IMCOST(12)=MNHOLE*MBCOST(12)
      IMCOST(13)=NLIGHT*MBCOST(13)
*
      IMCOST(14)=TSIGN*MBCOST(14)
      IMCOST(15)=NTRAFF*MBCOST(15)
*
      IMCOST(16)=TSING*MBCOST(16)
      IMCOST(17)=TSING*MBCOST(17)
```

```
      IMCOST(18)=TSING*MBCOST(18)
*
      IMCOST(19)=TMULT*MBCOST(19)
      IMCOST(20)=TMULT*MBCOST(20)
      IMCOST(21)=TMULT*MBCOST(21)
*
*  operation cost
      IOCOST(1)=WMPIPE*OBCOST(1)
      IOCOST(2)=DSPIPE*OBCOST(2)
      IOCOST(3)=SSPIPE*OBCOST(3)
*
      IOCOST(4)=TPAREA*OBCOST(4)
      IOCOST(5)=TWK*OBCOST(5)
      IOCOST(6)=TBR*OBCOST(6)
      IOCOST(7)=TRL*OBCOST(7)
      IOCOST(8)=TBL*OBCOST(8)
*
      IOCOST(9)=APARK*OBCOST(9)
      IOCOST(10)=ABUFF*OBCOST(10)
*
      IOCOST(11)=NBASIN*OBCOST(11)
      IOCOST(12)=MNHOLE*OBCOST(12)
      IOCOST(13)=NLIGHT*OBCOST(13)
*
      IOCOST(14)=TSIGN*OBCOST(14)
      IOCOST(15)=NTRAFF*OBCOST(15)
*
      IOCOST(16)=TSING*OBCOST(16)
      IOCOST(17)=TSING*OBCOST(17)
      IOCOST(18)=TSING*OBCOST(18)
*
      IOCOST(19)=TMULT*OBCOST(19)
      IOCOST(20)=TMULT*OBCOST(20)
      IOCOST(21)=TMULT*OBCOST(21)
*
*
*
      RETURN
      END
```

B.2.25 SUBROUTINE: OPERAT

```
      SUBROUTINE OPERAT

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER TYPE,METHOD,ELEM,OLIFE,WUNIT1,TSIGN,TIME
        &     ,ICDFLG,IDAFLG,TRIFLG,CCTFLG,OPRFLG,MNTFLG,ECNFLG,ACCFLG

      REAL IOCOST,OGRAD,OGRATE,OPCOST,OINST,OPPACC

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
        &     /AREA10/APARK,ABUFF,PCOST,BCOST
        &     /AREA17/COST25,COST26,COST27,COST28,COST29,COST30,COST31,
        &     COST32,COST33,MBCOST(25),OBCOST(25)
        &     /AREA20/ERRFLG,ICDFLG,IDAFLG,TRIFLG,CCTFLG,OPRFLG,MNTFLG,ECNFLG,
        &     /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &     /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
        &     ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
        &     /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOLE,
        &     MHCOST,NBASIN,CBCOST,NTRAFF,TFCOST,TWK,TBR,TRL,TBL,
        &     WKCOST,BRCOST,RLCOST,BLCOST
        &     /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &     ,MNPACC(25,10),OPPACC(25,10)
        &     /AREA47/NACCNT,ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &     ,DEFINE(100)
        &     /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &     ,OPCOST(25,3),OCAPAF(25,10,3)
        &     /AREA51/TPAREA,SSPIPE,WMPIPE,DSPIPE
*
      TIME=3
*
*
      DO 100 TYPE=1,21
*  determine which method is going to be used
            TPCENT=0.0
            DO 110 I=1,NACONT(TYPE)
                    TPCENT=TPCENT+PRCENT(TYPE,I)
110                 CONTINUE
            IF ( TPCENT.LE.1 ) THEN
*  using the ratio or percentage
              METHOD=1
            ELSE
*  the given cost are already broke down
              METHOD=2
            ENDIF
*
*  find the present, annual, and future worth
            IF ( OGRAD(TYPE).EQ.0.0 )  THEN
              CALL GOGRAD (IOCOST(TYPE),OGRATE(TYPE),OINST(TYPE),OLIFE(TYPE
        &                  ,P,A,F)
              OPCOST(TYPE,1)=P
              OPCOST(TYPE,2)=A
```

```
              OPCOST(TYPE,3)=F
            ELSE
              OPCOST(TYPE,1)=ARITHP(IOCOST(TYPE),OGRAD(TYPE),OINST(TYPE)
     &                ,OLIFE(TYPE))
              OPCOST(TYPE,2)=ARITHA(IOCOST(TYPE),OGRAD(TYPE),OINST(TYPE)
     &                ,OLIFE(TYPE))
              OPCOST(TYPE,3)=ARITHF(IOCOST(TYPE),OGRAD(TYPE),OINST(TYPE)
     &                ,OLIFE(TYPE))
            ENDIF
*
*  apply the correct method
            IF ( METHOD.EQ.1 )  THEN
              DO 120 I=1,NACONT(TYPE)
                OPPACC(TYPE,I)=IOCOST(TYPE)*PRCENT(TYPE,I)
                OCAPAF(TYPE,I,1)=OPCOST(TYPE,1)*PRCENT(TYPE,I)
                OCAPAF(TYPE,I,2)=OPCOST(TYPE,2)*PRCENT(TYPE,I)
                OCAPAF(TYPE,I,3)=OPCOST(TYPE,3)*PRCENT(TYPE,I)
                DO 130 J=1,NACCNT
                  IF ( ACCNUM(J).EQ.ACCONT(TYPE,I) )  THEN
                    ELEM=J
                  ENDIF
130                   CONTINUE
                AOCOST(ELEM)=AOCOST(ELEM)+OPPACC(TYPE,I)
120                 CONTINUE
            ELSE
              DO 140 I=1,NACONT(TYPE)
                DO 150 J=1,NACCNT
                  IF ( ACCNUM(J).EQ.ACCONT(TYPE,I) )  THEN
                    ELEM=J
                  ENDIF
150                   CONTINUE
                AOCOST(ELEM)=AOCOST(ELEM)+PRCENT(TYPE,I)
140                 CONTINUE
            ENDIF
100         CONTINUE
*
*  find the total operation cost
   TOCOST=0.0
   DO 160 I=1,21
            TOCOST=IOCOST(I)+TOCOST
160         CONTINUE

*
*
*  summary
*
*
   IF ( OPRFLG.EQ.0 )  THEN
   WRITE (WUNIT1,900)
   DO 257 I=1,TIME
            WRITE (WUNIT1,905)
257         CONTINUE
   WRITE (WUNIT1,910)
   DO 258 I=1,TIME
            WRITE (WUNIT1,911)
258         CONTINUE
   WRITE (WUNIT1,915)
```

```
   DO 259 I=1,TIME
          WRITE (WUNIT1,920)
259       CONTINUE
*
   WRITE (WUNIT1,925) WMPIPE,OBCOST(1),IOCOST(1)
   WRITE (WUNIT1,930) DSPIPE,OBCOST(2),IOCOST(2)
   WRITE (WUNIT1,935) SSPIPE,OBCOST(3),IOCOST(3)
   WRITE (WUNIT1,940) TPAREA,OBCOST(4),IOCOST(4)
   WRITE (WUNIT1,955) TWK,OBCOST(5),IOCOST(5)
   WRITE (WUNIT1,960) TBR,OBCOST(6),IOCOST(6)
   WRITE (WUNIT1,965) TRL,OBCOST(7),IOCOST(7)
   WRITE (WUNIT1,970) TBL,OBCOST(8),IOCOST(8)
   WRITE (WUNIT1,1063) APARK,OBCOST(9),IOCOST(9)
   WRITE (WUNIT1,1065) ABUFF,OBCOST(10),IOCOST(10)
   WRITE (WUNIT1,985) NBASIN,OBCOST(11),IOCOST(11)
   WRITE (WUNIT1,990) MNHOLE,OBCOST(12),IOCOST(12)
   WRITE (WUNIT1,1010) NLIGHT,OBCOST(13),IOCOST(13)
   WRITE (WUNIT1,1015) TSIGN,OBCOST(14),IOCOST(14)
   WRITE (WUNIT1,1020) NTRAFF,OBCOST(15),IOCOST(15)
   WRITE (WUNIT1,1030) INT(TSING)
   WRITE (WUNIT1,1035) OBCOST(16),IOCOST(16)
   WRITE (WUNIT1,1040) OBCOST(17),IOCOST(17)
   WRITE (WUNIT1,1045) OBCOST(18),IOCOST(18)
   WRITE (WUNIT1,1050) INT(TMULT)
   WRITE (WUNIT1,1035) OBCOST(19),IOCOST(19)
   WRITE (WUNIT1,1040) OBCOST(20),IOCOST(20)
   WRITE (WUNIT1,1045) OBCOST(21),IOCOST(21)
   WRITE (WUNIT1,1070) TOCOST
   DO 260 I=1,TIME
          WRITE (WUNIT1,1080) TOCOST
260       CONTINUE
   ENDIF
*  30,20,20,25
900       FORMAT ('1',' O P E R A T I O N   C O S T')
905       FORMAT ('+',' O P E R A T I O N   C O S T')
910       FORMAT (1X, ' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
911       FORMAT ('+',' ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
915       FORMAT (/////,6X,'DESCRIPTION',20X,'QUANTITY',17X,'UNIT COST',
     &    11X,'OPERATION COST')
920       FORMAT ('+',5X,'DESCRIPTION',20X,'QUANTITY',17X,'UNIT COST',
     &    11X,'OPERATION COST')
925       FORMAT (//,6X,'WATERMAIN',19X,F10.2,'m',6X,2(8X,'$',1X,F10.2,5X)
930       FORMAT (6X,'DOMESTIC SEWER',14X,F10.2,'m',6X,2(8X,'$',1X,F10.2,5
935       FORMAT (6X,'STORM SEWER',17X,F10.2,'m',6X,2(8X,'$',1X,F10.2,5X))
940       FORMAT (//,6X,'PAVEMENT',20X,F10.2,'sq. m',2X,2(8X,'$',1X,F10.2,
955       FORMAT (//,6X,'SIDEWALK',20X,F10.2,'sq. m',2X,2(8X,'$',1X,F10.2,
960       FORMAT (6X,'BARRIER CURB & GUTTER',7X,F10.2,'m',
     &    6X,2(8X,'$',1X,F10.2,5X))
965       FORMAT (6X,'ROLLED CURB & GUTTER',8X,F10.2,'m',
     &    6X,2(8X,'$',1X,F10.2,5X))
970       FORMAT (6X,'BLVD. CURB & GUTTER',9X,F10.2,'m',
     &    6X,2(8X,'$',1X,F10.2,5X))
985       FORMAT (//,6X,'CATCH-BASIN',20X,I4,2X,' units',
     &    2X,2(8X,'$',1X,F10.2,5X))
990       FORMAT (//,6X,'MANHOLE',24X,I4,2X,' units',
     &    2X,2(8X,'$',1X,F10.2,5X))
1010      FORMAT (//,6X,'STREET LIGHT',19X,I4,2X,' units',
```

```
     &      2X,2(8X,'$',1X,F10.2,5X))
1015        FORMAT (6X,'STREET SIGN',20X,I4,2X,' units',
     &      2X,2(8X,'$',1X,F10.2,5X))
1020        FORMAT (6X,'TRAFFIC SIGN',19X,I4,2X,' units',
     &      2X,2(8X,'$',1X,F10.2,5X))
1030        FORMAT (//,6X,'SINGLE FAMILY',18X,I4,2X,' units')
1035        FORMAT (10X,'- GAS',36X,2(8X,'$',1X,F10.2,5X))
1040        FORMAT (10X,'- POWER',34X,2(8X,'$',1X,F10.2,5X))
1045        FORMAT (10X,'- TELEPHONE',30X,2(8X,'$',1X,F10.2,5X))
1050        FORMAT (6X,'MULTI-FAMILY',19X,I4,2X,' units')
1063        FORMAT (//,6X,'PARK DEVELOPMENT',12X,F10.2,'Ha',
     &      5X,2(8X,'$',1X,F10.2,5X))
1065        FORMAT (6X,'BUFFER DEVELOPMENT',10X,F10.2,'Ha',
     &      5X,2(8X,'$',1X,F10.2,5X))
1070        FORMAT (//,11X,'T O T A L',64X,'$',1X,F10.2)
1080        FORMAT ('+',10X,'T O T A L',64X,'$',1X,F10.2)
*
*
   RETURN
   END
```

B.2.26 SUBROUTINE: OUTDAT1

```
*****************************************************************
*                                                               *
*                       SUBROUTINE OUTDAT1                      *
*                                                               *
*            This subroutine OUTPUTS the unit cost data.        *
*                                                               *
*****************************************************************
*
*    LOCAL VARIABLES
*
*  I,J  - LOOP CONTROL VARIABLES
*
   SUBROUTINE OUTDAT1
   INTEGER WUNIT1,WUNIT2,RUNIT1,RUNIT2
*
   COMMON
     &      /AREA11/COST1,COST2/AREA12/COST3,COST4/AREA13/COST5,COST6
     &      /AREA14/COST7,COST8,COST9,COST10
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA16/COST19,COST20,COST21,COST22,COST23,COST24
     &      /AREA17/COST25,COST26,COST27,COST28,COST29,COST30,COST31,
     &  COST32,COST33
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
*
   WRITE(WUNIT1,60)COST1,COST2,COST5,COST6,
     &      COST7,COST8,COST9,COST10,
     &      COST3,COST4,
     &      COST19,COST20,COST21,COST22,COST23,COST24
60 FORMAT('1UNIT PRICES',/X,11('='),///,X'DESCRIPTION'
     &      T37,'UNITS',T54,'UNIT PRICE',///
     &      X,'STREET SIGN',T32,'$/INTERSECTION',T53,F9.2,/
     &      X,'STREET LIGHT',T36,'$/LIGHT',T53,F9.2,//
     &      X,'COLLECTOR STREET',T36,'$/SQ M',T53,F9.2,/
     &      X,'LOCAL STREET',T36,'$/SQ M',T53,F9.2,//
     &      X,'SIDEWALK',T36,'$/SQ M',T53,F9.2,/
     &      X,'BARRIER CURB & GUTTER',T38,'$/M',T53,F9.2,/
     &      X,'ROLLED CURB & GUTTER',T38,'$/M',T53,F9.2,/
     &      X,'BLVD CURB & GUTTER',T38,'$/M',T53,F9.2,//
     &      X,'PARK DEVELOPMENT',T37,'$/HA',T53,F9.2,/
     &      X,'BUFFER DEVELOPMENT',T37,'$/HA',T53,F9.2,//
     &      X,'POWER (SINGLE FAMILY)',T36,'$/UNIT',T53,F9.2,/
     &      X,'      (MULTI  FAMILY)',T36,'$/UNIT',T53,F9.2,/
     &      X,'GAS   (SINGLE FAMILY)',T36,'$/UNIT',T53,F9.2,/
     &      X,'      (MULTI  FAMILY)',T36,'$/UNIT',T53,F9.2,/
     &      X,'PHONE (SINGLE FAMILY)',T36,'$/UNIT',T53,F9.2,/
     &      X,'      (MULTI  FAMILY)',T36,'$/UNIT',T53,F9.2,//)
   WRITE(WUNIT1,61)COST25,COST26,COST27,COST28,
     &      COST29,COST30,COST31,COST32,COST33
61 FORMAT(X,'STREET SWEEPING',T36,'$/KM/YR',T53,F9.2,/
     &      X,'WINTER ROAD',T36,'$/KM/YR',T53,F9.2,/
     &      X,'ASPHALT MAINTENANCE',T36,'$/KM/YR',T53,F9.2,/
     &      X,'CONCRETE MAINTENANCE',T36,'$/KM/YR',T53,F9.2,//
     &      X,'SEWER MAINTENANCE',T36,'$/KM/YR',T53,F9.2,/
     &      X,'WATER MAINTENANCE',T36,'$/KM/YR',T53,F9.2,//
```

```
     &      X,'SOLID WASTE COLLECTION',T36,'$/HA/YR',T53,F9.2,/
     &      X,'STREET LIGHT MAINTENANCE',T35,'$/LIGHT/YR',T53,F9.2,/
     &      X,'PARK & BUFFER MAINTENANCE',T36,'$/HA/YR',T53,F9.2,//)
   WRITE(WUNIT1,62)COST16,COST17,COST18,COST14,COST15
62 FORMAT(X,'CATCHBASIN'/
     &      X,'  BASE, FRAME, COVER, ETC.',T33,'$/CATCHBASIN',T53,F9.2,/
     &      X,'  SHAFT',T35,'$/M-depth',T53,F9.2,/
     &      X,'  LEAD        DEPTH INDEX = 1',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 2',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 3',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 4',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 5',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 6',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 7',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 8',T38,'$/M',T53,F9.2,/
     &      X,'              DEPTH INDEX = 9',T38,'$/M',T53,F9.2,//
     &      X,'MANHOLE'/
     &      X,'  BASE, FRAME, COVER, ETC.',T35,'$/MANHOLE',T53,F9.2,/
     &      X,'  SHAFT',T35,'$/M-depth',T53,F9.2,//)
   WRITE(WUNIT1,63)((I,COST11(I)),I=1,6)
63 FORMAT('1 WATERMAIN @ 2.44 M OF COVER'/,
     &      6('            DIAMETER INDEX ='I2,T38,'$/M'T52,F9.2,/))
   WRITE(WUNIT1,64)((I,(COST12(J,I),J=1,9)),I=1,5)
64 FORMAT(//X,'DOMESTIC SEWER ($/M)',/,T69,'DEPTH INDEX',/,
     &      T34'1'T44'2'T54'3'T64'4'T74'5'T84'6'T94'7'T104'8'
     &      T114'9'//5(T4'DIAMETER INDEX ='I2,T28,F9.2,T38,F9.2,T48,F9.2,T58
     &      ,F9.2,T68,F9.2,T78,F9.2,T88,F9.2,T98,F9.2,T108,F9.2,/))
   WRITE(WUNIT1,65)((I,(COST13(J,I),J=1,9)),I=1,14)
65 FORMAT(//X,'STORM SEWER ($/M)',/,T69,'DEPTH INDEX',/,
     &      T34'1'T44'2'T54'3'T64'4'T74'5'T84'6'T94'7'T104'8'
     &      T114'9'//14(T4'DIAMETER INDEX ='I3,T28,F9.2,T38,F9.2,T48,F9.2,T5
     &      ,F9.2,T68,F9.2,T78,F9.2,T88,F9.2,T98,F9.2,T108,F9.2,/))
   RETURN
   END
```

B.2.27 **SUBROUTINE: OUTDAT2**

```
*****************************************************************
*                                                               *
*                      SUBROUTINE OUTDAT2                       *
*                                                               *
*       This subroutine OUTPUTS the Subdivision Design Data.    *
*                                                               *
*****************************************************************
*
*    LOCAL VARIABLES
*
*  L,N  - LOOP CONTROL VARIABLES
*              where: L refers to the link number.
*                     N refers to the node number.
*
   SUBROUTINE OUTDAT2
*
   PARAMETER
     &      NODNUM=100,LINNUM=150,INTNUM=10
   INTEGER
     &  TROAD,TBTRYW,TBTRYD,TBTRYS,WUNIT1,RUNIT1,RUNIT2,WUNIT2
   REAL
     &  LLINK,LROAD,LENGTH,MULTI,INTRST
   COMMON
     &      /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA5/WWALK(LINNUM),BARIER(LINNUM),ROLLED(LINNUM),BLVD(LINNUM)
     &      /AREA6/NWMLNK,WLINKS(LINNUM),NWMGRP,TBTRYW(LINNUM,4)
     &      /AREA7/NDSLNK,DLINKS(LINNUM),NDSGRP,TBTRYD(LINNUM,4)
     &      /AREA8/NSSLNK,SLINKS(LINNUM),NSSGRP,TBTRYS(LINNUM,4)
     &      /AREA9/IQ,L,P(LINNUM),PDS(LINNUM),ASS(LINNUM)
     &      /AREA10/APARK,ABUFF
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2

   WRITE(WUNIT1,161)
161         FORMAT('1LINK DATA',/X,9('='),///3X,'TYPE 1 = COLLECTOR STREET'
     &      ,/8X,'2 = LOCAL STREET',//X,
     &      T35,'FAMILY UNITS',T58,'STREET',T84,'CURB & GUTTER LENGTH',/
     &      T26,'LINK',T34,14('-'),T51,21('-'),T75,'WALK',T83,23('-'),/X
     &      'LINK   NODE1   NODE2   LENGTH   SINGLE   MULTI   LENGTH   '
     &      'WIDTH   TYPE   WIDTH   BARRIER   ROLLED   BLVD',/
     &      T2,'(#)',T10,'(#)',T18,'(#)',T26,'(M)',T35,'(#)',T44,'(#)',
     &      T52,'(M)',T61,'(M)',T76,'(M)',T85,'(M)',T94,'(M)',T102,'(M)',/)


   WRITE(WUNIT1,162)((L,NODE1(L),NODE2(L),LLINK(L),UNITS(L),MULTI(L),
     &      LROAD(L),WROAD(L),TROAD(L),WWALK(L),BARIER(L),ROLLED(L),
     &      BLVD(L)),L=1,NLINK)
162         FORMAT(I4,T9,I4,T17,I4,T24,F7.2,T35,F4.0,T43,F4.0,T50,F7.1,
     &      T58,F7.1,T68,I2,T75,F4.1,T83,F7.1,T92,F7.1,T99,F7.1)


   WRITE(WUNIT1,163)
```

```
163         FORMAT('1NODE DATA',/X,9('='),///3X,
     &      'TYPE 1 = CUL DE SAC, DEAD-END, TERMINATION',/8X,
     &      '2 = DIRECTIONAL',/8X,'3 = 3-WAY STREET INTERSECTION',/8X,
     &      '4 = 4-WAY STREET INTERSECTION',/8X,'5 = OTHER',///X,
     &      ' NODE   TYPE   ELEV',/2X,'(#)',T17,'(M)',/)
   WRITE(WUNIT1,164)((N,ABS(NTYPE(N)),ELEV(N)),N=1,NNODE)
164         FORMAT(X,I4,T10,I2,T15,F6.2)

   RETURN

   END
```

B.2.28 SUBROUTINE: PARKBUFF

```
      SUBROUTINE PARKBUFF
*****************************************************************
*                                                               *
*                    PARK & BUFFER REQUIREMENTS                 *
*                                                               *
*****************************************************************
*
*  PCOST  - total cost of park development
*  BCOST  - total cost of buffer development
*
*
*
      PARAMETER
     &      NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &      RUNIT1,RUNIT2,WUNIT1,WUNIT2

      COMMON
     &      /AREA10/APARK,ABUFF,PCOST,BCOST
     &      /AREA12/COST3,COST4
     &      /AREA19/TLCOST
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2

      PCOST=APARK*COST3
      BCOST=ABUFF*COST4
*
      CALL ASPLIT(9,PCOST)
      CALL ASPLIT(10,BCOST)
*
      TLCOST=TLCOST+PCOST+BCOST
*  WRITE(WUNIT1,910)APARK,PCOST,ABUFF,BCOST
910         FORMAT(10X,'PARK DEVELOPMENT (Ha)',T41,F7.1,T54,F7.0,/,
     &  10X,'BUFFER DEVELOPMENT (Ha)',T41,F7.1,T54,F7.0,/)
*  WRITE(WUNIT1,920)TLCOST
920         FORMAT(36X,'T O T A L',T52,F9.0,//)
*
*
*
      END
```

B.2.29 SUBROUTINE: PAVEMENT

```
      SUBROUTINE PAVEMENT

*****************************************************************
*                                                               *
*                        SUBROUTINE  PAVEMENT                   *
*                                                               *
*****************************************************************
*
*
*  XXXXXX -
*  ATTLIN - an array describing the links attached to a given node.
*  ALINK  - the area of the road on a link.
*  AINTS  - area of intersection.
*  CORWD  - corresponding road width of the given link.
*  LROAD  - length of the road on a link (calculated).
*  CNODE  - current node in the memory.
*  PNODE  - previous node in the memory.
*  CLINK  - current link in the memory.
*  BULB   - this flag indicates the number of bulb curve(s) found.
*  DIRECT - a flag indicates the origin/dest node for a given link.
*  WD     - road width ( a value return from the ADJ routine ).
*  LEN    - road length ( a value return from the COR/SAC/TUBE routine ).
*  ASAC   - area of the road ( a value return from the COR/SAC/TUBE routine
*  TAREA  - total area of the pavement.
*  TLEN   - total length of the pavement.
*  LOCLEN - total length of the pavement of the local street only.
*  COLLEN - total length of the pavement of the collector street only.
*  ACOL   - total area of the pavement of the collector street only.
*  ALOC   - total area of the pavement of the local street only.
*  TROAD  - type of road.
*  WROAD  - width of the road.
*  EXIT   - a flag indicates the exit node.
*  LLINK  - length of the link.
*  LSCOST - cost for the local street.
*  CSCOST - cost for the collector street.
*  RCURB  - radius of the curb.
*  I,N,J  - loop counter.


      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,CNODE,CLINK,
     &      BULB,ATTLIN,TROAD,PNODE,EXIT,LINK(3)

      REAL
     &  LLINK,LROAD,LOCLEN,LEN,LSCOST

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      RCURB
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
```

```
     &      /AREA13/COST5,COST6
     &   /AREA19/TLCOST
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &      ,PVCOST(LINNUM),ACOL,ALOC,COLLEN,LOCLEN
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA31/TLEN,TAREA,RDPINT(LINNUM)
*
*
*
*  define pi
   PI=ACOS(-1.0)
*
*
*
*  clear the counter and total to zero
   TAREA=0.0
   TLEN=0.0
   LOCLEN=0.0
   COLLEN=0.0
   ACOL=0.0
   ALOC=0.0
*
*
*
*  determine the corresponding road width for every link
   DO 101 I=1,NLINK
*          ORIGIN NODE
           CNODE=NODE1(I)
           IF ( NTYPE(CNODE).EQ.3 .OR. NTYPE(CNODE).EQ.4 )  THEN
             CALL ADJ(CNODE,I,WD)
             CORWD(I,1)=WD
           ELSE
             CORWD(I,1)=0.0
           ENDIF
*          DESTINATION NODE
111                CNODE=NODE2(I)
           IF ( NTYPE(CNODE).EQ.3 .OR. NTYPE(CNODE).EQ.4 ) THEN
             CALL ADJ(CNODE,I,WD)
             CORWD(I,2)=WD
           ELSE
                   CORWD(I,2)=0.0
           ENDIF
101        CONTINUE
*
*
*
*          DECISION MAKER
   DO 121 I=1,NLINK
*  define the origin node (N1) and dest. node (N2)
           N1=NODE1(I)
           N2=NODE2(I)
*  EASEMENT
   IF ( WROAD(I).EQ.0.0 )  GOTO 121
*  DEAD END/CUL DE SAC/TURE SAC
   IF ( NTYPE(N1).EQ.1 .OR. NTYPE(N2).EQ.1 )  THEN
           IF ( NTYPE(N1).EQ.1 )   THEN
```

```
              CNODE=N2
              PNODE=N1
            ELSE
              CNODE=N1
              PNODE=N2
            ENDIF
            IF  ( NTYPE(CNODE).EQ.2 )  THEN
              IF ( RADIUS(PNODE).NE.0.0 )  THEN
                CALL SAC(PNODE,I,ASAC,LEN)
                ALINK(I)=ASAC
                LROAD(I)=LEN
                RDPINT(I)=LROAD(I)
              ELSE
                ALINK(I)=LLINK(I)*WROAD(I)
                LROAD(I)=LLINK(I)
                RDPINT(I)=LROAD(I)
              ENDIF
            ELSE
*           TUBE SAC
              IF ( RADIUS(PNODE).NE.0.0 )  THEN
                CALL TUBE(PNODE,I,ASAC,LEN)
                ALINK(I)=ASAC
                LROAD(I)=LEN
                RDPINT(I)=LROAD(I)
              ELSE
                TV=LLINK(I)-(CORWD(I,1)+CORWD(I,2))/2.0
                LROAD(I)=TV
                ALINK(I)=TV*WROAD(I)
                RDPINT(I)=LROAD(I)
              ENDIF
            ENDIF
    ENDIF
*   STRAIGHT ROAD/STREET AND BULB CURVE
*   set the bulb curve flag to zero
    BULB=0
    IF ( NTYPE(N1).EQ.2.AND.NTYPE(N2).EQ.2 ) THEN
            IF ( RADIUS(N1).NE.0.0 )  THEN
*           BULB CURVE AT N1
              CNODE=N1
              BULB=1
            ENDIF
            IF ( RADIUS(N2).NE.0.0 )  THEN
*           BULB CURVE AT N2
              CNODE=N2
              BULB=BULB+1
            ENDIF
            IF ( BULB.NE.0 )  THEN
              CALL COR(CNODE,I,BULB,ASAC,LEN)
              ALINK(I)=ASAC
              LROAD(I)=LEN
              RDPINT(I)=LROAD(I)
            ELSE
              ALINK(I)=LLINK(I)*WROAD(I)
              LROAD(I)=LLINK(I)
              RDPINT(I)=LROAD(I)
            ENDIF
    ENDIF
```

```
*   STREET INTERSECTION/DIRECTIONAL NODE
*   set the bulb curve flag to zero
    BULB=0
    IF ( NTYPE(N1).EQ.2 .OR. NTYPE(N2).EQ.2 )  THEN
            IF ( NTYPE(N1).EQ.2 )  THEN
              PNODE=N1
              CNODE=N2
            ELSE
              PNODE=N2
              CNODE=N1
            ENDIF
            IF ( NTYPE(CNODE).EQ.3 .OR. NTYPE(CNODE).EQ.4 )  THEN
              IF ( RADIUS(PNODE).NE.0.0 )  THEN
                BULB=1
                CALL COR(PNODE,I,BULB,ASAC,LEN)
                EXTRA=(CORWD(I,1)+CORWD(I,2))/2.0
                ALINK(I)=ASAC-WROAD(I)*EXTRA
                LROAD(I)=LEN-EXTRA
                RDPINT(I)=LROAD(I)
              ELSE
                TV=LLINK(I)-(CORWD(I,1)+CORWD(I,2))/2.0
                ALINK(I)=TV*WROAD(I)
                LROAD(I)=TV
                RDPINT(I)=LROAD(I)
              ENDIF
            ENDIF
    ENDIF

*   INTERSECTION/INTERSECTION
    IF ( NTYPE(N1).EQ.3.OR.NTYPE(N1).EQ.4 .AND. NTYPE(N2).EQ.3
      &      .OR.NTYPE(N2).EQ.4 )  THEN
            LROAD(I)=LLINK(I)-(CORWD(I,1)+CORWD(I,2))/2.0
            ALINK(I)=LROAD(I)*WROAD(I)
            RDPINT(I)=LROAD(I)
    ENDIF
121         CONTINUE
*
*
*
*   determine the area for every intersection
*   and distribute the calculated area to the collector
*   street and major local street
*   define the area in front of the side-walk curve
    CURVE=RCURB**2.0*(1.0-0.25*PI)
*
    DO 131 N=1,NNODE
            IF ( NTYPE(N).EQ.3 .OR. NTYPE(N).EQ.4 )  THEN
              IF ( NTYPE(N).EQ.3 )  THEN
                A=2.0
              ELSE
                A=4.0
              ENDIF
              CALL DMAKER (N,2,LINK)
              CLINK=LINK(1)
              IF ( EXIT(N).EQ.1 )  THEN
                AINTS(N)=(WROAD(CLINK)**2.0)/2.0+2.0*CURVE
                ALINK(CLINK)=ALINK(CLINK)+AINTS(N)
```

```
                RDPINT(CLINK)=RDPINT(CLINK)+0.5*CORWD(CLINK,DIRECT)
              ELSE
                IF ( NODE1(CLINK).EQ.N )  THEN
                  DIRECT=1
                ELSE
                  DIRECT=2
                ENDIF
                AINTS(N)=WROAD(CLINK)*CORWD(CLINK,DIRECT)+A*CURVE
                DO 135 J=1,2
                  ALINK(LINK(J))=ALINK(LINK(J))+AINTS(N)/2.0
                  RDPINT(LINK(J))=RDPINT(LINK(J))+0.5*CORWD(CLINK,DIRECT)
135                     CONTINUE
              ENDIF
            ENDIF
131         CONTINUE
*
*
*
*  ADD UP THE AREA/LENGTH FOR THE COLLECTOR/LOCAL STREET
   DO 141 I=1,NLINK
            IF ( TROAD(I).EQ.1 )  THEN
              ACOL=ACOL+ALINK(I)
              COLLEN=COLLEN+LROAD(I)
              PVCOST(I)=ALINK(I)*COST5
            ELSE
              ALOC=ALOC+ALINK(I)
              LOCLEN=LOCLEN+LROAD(I)
              PVCOST(I)=ALINK(I)*COST6
            ENDIF
141         CONTINUE
*
*
*
*  add up the total area and length in the sub-division
   TAREA=ACOL+ALOC
   TLEN=COLLEN+LOCLEN
*
*
*
*  calculate the costs
   CSCOST=ACOL*COST5
   LSCOST=ALOC*COST6
   TLCOST=TLCOST+CSCOST+LSCOST
*
*
*  OUTPUT THE RESULT  ( SPECIAL )
*  ROAD AND AREA

   WRITE (088,840)
   WRITE (088,850)
   WRITE (088,860) TLEN,TAREA
   WRITE (088,865) COLLEN,ACOL,LOCLEN,ALOC
   WRITE (088,870)

   DO 365 I=1,NLINK
   WRITE (088,880) I,ALINK(I),LROAD(I)
365         CONTINUE
```

```
      DO 366 I=1,NLINK
      WRITE (088,966)I,CORWD(I,1),CORWD(I,2)

366         CONTINUE

966         FORMAT(1X,'CORWD :'I3,2(3X,F9.3))
800         FORMAT (' ',' I N T E R S E C T I O N S')
805         FORMAT (' ',' -------------------------')
810         FORMAT (////,' Three ways :',2X,I3,/,' Four ways :',2X,I3)
820         FORMAT (////,' NODE',20X,' AREA')
830         FORMAT (' ',I3,18X,F11.3)
840         FORMAT (////,' R O A D   A N D   A R E A')
850         FORMAT (' ','-------------------------')
860         FORMAT (////,' TOTAL road length :',2X,F11.3,
     &/,' TOTAL pavement :',2X,F11.3)
870         FORMAT (////,' LINK',21X,' AREA',23X,' LENGTH')
880         FORMAT (' ',I3,2(19X,F11.3))
865         FORMAT (//,' COLLECTOR ST. LENGTH :',2X,F11.3,3X,'AREA :',2X,F11
     &/,' LOCAL ST. LENGTH :',2X,F11.3,3X,'AREA :',2X,F11.3)


      RETURN
      END
```

B.2.30 SUBROUTINE: SIDEWALK

```
      SUBROUTINE SIDEWALK


*****************************************************************
*                                                               *
*                      SIDEWALK REQUIREMENT                     *
*                                                               *
*****************************************************************
*
*
*
*  SRCOST - total cost of the sidewalk requirement per link
*  WKCOST - total cost of sidewalk
*  BRCOST - total cost of barrier curb & gutter
*  RLCOST - total cost of rolled curb & gutter
*  BLCOST - total cost of blvd curb & gutter
*  LWALK  - length of the sidewalk (m)
*  WWALK  - width of the sidewalk (m)
*  BARIER - length of barrier curb & gutter (m)
*  ROLLED - length of rolled curb & gutter (m)
*  BLVD   - length of blvd curb & gutter (m)
*  WIDTH  - min. width of the sidewalk
*  ASIDE  - area of sidewalk per link (sq. m)
*  TWK    - total area of sidewalk (sq. m)
*  TBR    - total length of barrier curb & gutter (m)
*  TRL    - total length of rolled curb & gutter (m)
*  TBL    - total length of blvd curb & gutter (m)
*  TROAD  - type of road
*
*
*
      PARAMETER NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &     WUNIT1,WUNIT2,RUNIT1,RUNIT2,ERRFLG

      REAL
     &     LWALK,WIDTH(2)

      COMMON
     &     /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &     UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &     /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &     /AREA5/LWALK(LINNUM),WWALK(LINNUM),BARIER(LINNUM),ROLLED(LINNUM)
     &     ,BLVD(LINNUM)
     &     /AREA14/COST7,COST8,COST9,COST10
     &     /AREA19/TLCOST
     &     /AREA20/ERRFLG
     &     /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &     /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
     &     ASUB,ASIDE(LINNUM),ASING,TSING,TMULT,APAVE,AWALK
     &     /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,TSIGN,CSIGN,MNHOL
     &     MHCOST,NBASIN,CBCOST,NTRAFF,TRACOS,TWK,TBR,TRL,TBL,
     &     WKCOST,BRCOST,RLCOST,BLCOST
```

```
     &      /AREA45/WKCOSL(LINNUM),BRCOSL(LINNUM),RLCOSL(LINNUM)
     &      ,BLCOSL(LINNUM)
*
*
*
*  if the length of the sidewalk is equal to zero, then the street
*  will have no sidewalk
*  if the width of the sidewalk is equal to zero, then use the min.
*  value for the width of the sidewalk.
*
*  initialize the min. width of the sidewalk
   WIDTH(1)=1.5
   WIDTH(2)=1.2
*
*  set total length to zero
   TWK=0.0
   TBR=0.0
   TRL=0.0
   TBL=0.0
*
*
*
   DO 100 L=1,NLINK
*
*  set the total cost of the sidewalk requirement per link to zero
           SRCOST(L)=0.0
*
           IF ( WWALK(L).EQ.0 )  THEN
             ASIDE(L)=LWALK(L)*WIDTH(TROAD(L))
           ELSE
             ASIDE(L)=LWALK(L)*WWALK(L)
           ENDIF
*
           TWK=TWK+ASIDE(L)
           TBR=TBR+BARIER(L)
           TRL=TRL+ROLLED(L)
           TBL=TBL+BLVD(L)
           WKCOSL(L)=ASIDE(L)*COST7
           BRCOSL(L)=BARIER(L)*COST8
           RLCOSL(L)=ROLLED(L)*COST9
           BLCOSL(L)=BLVD(L)*COST10
           SRCOST(L)=ASIDE(L)*COST7+BARIER(L)*COST8+ROLLED(L)*COST9+
     &             BLVD(L)*COST10
*
100        CONTINUE
*
*
*
   WKCOST=TWK*COST7
   BRCOST=TBR*COST8
   RLCOST=TRL*COST9
   BLCOST=TBL*COST10
*
   CALL ASPLIT(5,WKCOST)
   CALL ASPLIT(6,BRCOST)
   CALL ASPLIT(7,RLCOST)
   CALL ASPLIT(8,BLCOST)
```

```
*
   TLCOST=TLCOST+WKCOST+BRCOST+RLCOST+BLCOST
*
*  WRITE(WUNIT1,200)TWK,WKCOST,TBR,BRCOST
200         FORMAT(10X,'SIDEWALK (SQ.M)',T40,F7.0,T54,F7.0,/,
     &  10X,'BARRIER CURB & GUTTER (M)',T40,F7.0,T54,F7.0)
*  WRITE(WUNIT1,210)TRL,RLCOST,TBL,BLCOST
210         FORMAT(10X,'ROLLED CURB & GUTTER (M)',T40,F7.0,T54,F7.0,/,
     &  10X,'BLVD CURB & GUTTER (M)',T40,F7.0,T54,F7.0,/)
   END
```

B.2.31 **SUBROUTINE: SIGN**

```
      SUBROUTINE SIGN


*****************************************************************
*                                                               *
*                    STREET/TRAFFIC SIGNS REQUIREMENT           *
*                                                               *
*****************************************************************
*
*
*
*  NSTR    - total number of street sign in the sub-division
*  NSTOP   - number of STOP sign per node
*  NYELD   - number of YIELD sign per node
*  NCOL    - number of collector streets associate with the intersection
*  NLOC    - number of local streets associate with the intersection
*  CLINK   - current link in the memory
*  TYELD   - total number of YIELD sign in the sub-division
*  TSTOP   - total number of STOP sign in the sub-division
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,CLINK,TYELD,TSTOP,TROAD,ATTLIN
     &      ,LINK(3)

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      RCURB
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA11/COST1,COST2
     &      /AREA12/COST3,COST4
     &      /AREA13/COST5,COST6
     &      /AREA19/TLCOST
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA25/SPAMAN,SPALIT,SPACAT,FRATIO
     &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
     &      /AREA38/NYELD(NODNUM),NSTOP(NODNUM),TYELD,TSTOP,NSIGN(LINNUM),
     &      LYELD(LINNUM),LSTOP(LINNUM),SGCOST(LINNUM),YDCOST(LINNUM),
     &      STCOST(LINNUM)
     &      /AREA43/SSCOST,DSCOST,WMCOST,NLIGHT,LTCOST,NSTR,SNCOST,MNHOLE,
     &      MHCOST,NBASIN,CBCOST,NTRAFF,TFCOST,TWK,TBR,TRL,TBL,
     &      WKCOST,BRCOST,RLCOST,BLCOST
*
*
*
*  set the total to zero
```

```
      NSTR=0
      TYELD=0
      TSTOP=0
      YCOST=0.0
      SCOST=0.0
*
*  set the traffic signs per node to zero
      DO 90 I=1,NNODE
              NSTOP(I)=0
              NYELD(I)=0
90 CONTINUE
*
*  one street sign per intersection
*
*  when local/local streets cross each other, no traffic signs require
*
*  THREE WAYS :
*  when local/collector streets cross each other, 1 YEILD sign requires
*  when collector/collector streets cross each other, 1 STOP sign requires
*
*  FOUR WAYS:
*  when local/collector streets cross each other, 2 YEILD signs require
*  when collector/collector streets cross each other, 2 STOP signs require
*
      DO 100 I=1,NNODE
              IF ( NTYPE(I).EQ.3 .OR. NTYPE(I).EQ.4 )  THEN
                NSTR=NSTR+1
                CALL DMAKER (I,1,LINK)
                NSIGN(LINK(1))=NSIGN(LINK(1))+1
                SGCOST(LINK(1))=SGCOST(LINK(1))+COST1
*  set the collector/local street counter to zero
                NCOL=0
                NLOC=0
*
                DO 110 J=1,4
                  CLINK=ATTLIN(I,J)
                  IF ( CLINK.EQ.0 .OR. WROAD(CLINK).EQ.0.0 )  GOTO 110
                  IF ( TROAD(CLINK).EQ.1 )  THEN
*             collector street
                    NCOL=NCOL+1
                  ELSE
*             local street
                    NLOC=NLOC+1
                  ENDIF
110                     CONTINUE
*
                IF ( NCOL.GE. 2 )  THEN
*  local streets cross collector streets
                  IF ( NCOL.EQ.2 .AND. NLOC.LE.2 )  THEN
                    NYELD(I)=NTYPE(I)-2
                    CALL DMAKER2(I,NYELD(I),1,0,LINK)
                    DO 120 J=1,NYELD(I)
                      LYELD(LINK(J))=LYELD(LINK(J))+1
                      YDCOST(LINK(J))=YDCOST(LINK(J))+COST1
                      YCOST=YCOST+COST1
120                         CONTINUE
                  ENDIF
```

```
*  collector streets cross collector streets
                IF ( NCOL.GE.3. .AND. NLOC.EQ.0 )  THEN
                  NSTOP(I)=NTYPE(I)-2
                  CALL DMAKER2(I,NSTOP(I),0,1,LINK)
                  DO 130 J=1,NSTOP(I)
                    LSTOP(LINK(J))=LSTOP(LINK(J))+1
                    STCOST(LINK(J))=STCOST(LINK(J))+COST1
                    SCOST=SCOST+COST1
130                       CONTINUE
                ENDIF
*  uneven crossing with 3 collector streets
                IF ( NCOL.EQ.3. .AND. NLOC.NE.0 )  THEN
                  NSTOP(I)=2
                  CALL DMAKER2(I,NSTOP(I),1,1,LINK)
                  DO 140 J=1,NSTOP(I)
                    LSTOP(LINK(J))=LSTOP(LINK(J))+1
                    STCOST(LINK(J))=STCOST(LINK(J))+COST1
                    SCOST=SCOST+COST1
140                       CONTINUE
                ENDIF
              ELSE
*  uneven crossing with 1 collector street
                IF ( (NLOC+NCOL).GE.3 .AND. NCOL.NE.0 )  THEN
                  NYELD(I)=NTYPE(I)-2
                  CALL DMAKER2(I,NYELD(I),1,0,LINK)
                  DO 150 J=1,NYELD(I)
                    LYELD(LINK(J))=LYELD(LINK(J))+1
                    YDCOST(LINK(J))=YDCOST(LINK(J))+COST1
                    YCOST=YCOST+COST1
150                       CONTINUE
                ENDIF
              ENDIF
            ENDIF
100         CONTINUE
*
*
*
*  add up the total of the traffic signs required
   DO 160 I=1,NNODE
            TYELD=TYELD+NYELD(I)
            TSTOP=TSTOP+NSTOP(I)
160         CONTINUE
*
   TFCOST=YCOST+SCOST
   NTRAFF=TYELD+TSTOP
*
*
   SNCOST=NSTR*COST1
*
   CALL ASPLIT(14,SNCOST)
   CALL ASPLIT(15,TFCOST)
   TLCOST=TLCOST+SNCOST+TFCOST
*  WRITE(WUNIT1,710)NSTR,SNCOST
*  WRITE(WUNIT1,720)TSTOP,YCOST
*  WRITE(WUNIT1,730)TYELD,SCOST
*
*
```

```
*
710         FORMAT(10X,'STREET SIGNS (#)',T40,I6,T54,F7.0)
720         FORMAT(10X,'STOP   SIGNS (#)',T40,I6,T54,F7.0)
730         FORMAT(10X,'YEILD  SIGNS (#)',T40,I6,T54,F7.0,/)


*  OUTPUT THE RESULT (SPECIAL)

   WRITE (088,230)
   WRITE (088,235)
   WRITE (088,200) NSTR
   WRITE (088,210) TYELD
   WRITE (088,220) TSTOP
   WRITE (088,250)

   DO 180 I=1,NNODE

            IF ( NTYPE(I).EQ.3 .OR. NTYPE(I).EQ.4 )  THEN
                    WRITE (088,240) I,NYELD(I),NSTOP(I)
            ENDIF

180         CONTINUE


200         FORMAT (///,' NUMBER OF STREET SIGN -->',3X,I3)
210         FORMAT (/,' NUMBER OF YEILD SIGN -->',3X,I3)
220         FORMAT (/,' NUMBER OF STOP SIGN -->',3X,I3,///)
230         FORMAT ('1',' S I G N')
235         FORMAT (' -------')
250         FORMAT (' ','NODE',20X,'YEILD',20X,'STOP',/)
240         FORMAT (' ',1X,I3,22X,I3,20X,I3)


   RETURN
   END
```

B.2.32 **SUBROUTINE: STORM**

```
      SUBROUTINE STORM(SSCOST)
*****************************************************************
*                                                               *
*                        SUBROUTINE STORM                       *
*                                                               *
*  This subroutine designs the STORM SEWER system. For each link,  *
*  the subroutine: (1) estimates the generated storm sewer flow,   *
*  (2) calculates the slope and the pipe size required to carry    *
*  the estimated sewer flow, (3) selects a standard sized storm    *
*  sewer pipe based on the calculated diameter, (4) verifies the   *
*  maximum and minimum allowable water velocites, (5) adjusts the  *
*  pipe slope and/or the pipe size to meet the tolerances set by   *
*  the maximum and minimum allowable water velocites, and (6)      *
*  calculates the average depth of ground cover over the storm     *
*  sewer.                                                          *
*                                                                  *
*****************************************************************
*
*  SSDPTH   - MINIMUM DEPTH OF STORM SEWER (M)
*  SSCOST   - TOTAL COST OF STORM SEWER (M)
*  TSAREA   - TOTAL DRAINAGE AREA OF EACH DESTINATION LINK (HA)
*  SSLOPE   - SLOPE OF STORM SEWER FORM ORIGIN NODE TO DESTINATION
*             NODE
*  SSELEV   - ELEVATION OF STORM SEWER PIPE (M)
*  A2       - TOTAL DRAINAGE AREA OF EACH LINK (HECTARES)
*  QSS      - TOTAL GENERATED STORM SEWER FLOW (CMS)
*  D        - CALCULATED DIAMETER OF PIPE (M)
*  D3       - REQUIRED DIAMETER OF STORM SEWER (M)
*  DIASS    - DIAMETER OF STORM SEWER (mm)
*  INDX3    - DIAMETER INDEX FOR STORM SEWER (1 - 14)
*  DINDX3   - DEPTH INDEX FOR STORM SEWER (1 - 9)
*
      PARAMETER NODNUM=100,LINNUM=150,INTNUM=10
      INTEGER DINDX3,TBTRYS,WUNIT1,WUNIT2,RUNIT1,RUNIT2,CONST,ERRFLG
      REAL LLINK
      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA8/NSSLNK,SLINKS(LINNUM),NSSGRP,TBTRYS(LINNUM,4)
     &      /AREA9/IQ,L,P(LINNUM),PDS(LINNUM),ASS(LINNUM)
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
     &      /AREA19/TLCOST
     &      /AREA20/ERRFLG
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA22/PEOPLE,WVMIN,FRICTN,USE,FLOAD
     &      /AREA24/SSCOVR,SSVMIN,SSVMAX

      SSCOST=0.0
      WRITE(WUNIT1,230)SSVMIN,SSVMAX,SSCOVR
230         FORMAT('1STORM SEWER',/X,11('='),//,
     &      3X,'MINIMUM ACCEPTABLE PIPE VELOCITY (M/S) = ',F7.3,/
```

```
     &      3X,'MAXIMUM ACCEPTABLE PIPE VELOCITY (M/S) = ',F7.3,//,
     &      3X,'MINIMUM DEPTH OF GROUND COVER OVER PIPE (M) = ',F7.3,
     &      //T83'DEPTH OF EXCAVATION',/,T10,'TRIBUTARY',T36,
     &      'GROUND',T46,'CALCULATED',T72,'PIPE',T83,'------------------
     &      ,T107'AVERAGE'/' LINK'T9'POPULATION   FLOW RATE     SLOPE'
     &      T47'DIAMETER'T59'VELOCITY'T72'SLOPE'T84'NODE1'T96'NODE2'T108
     &      'DEPTH'T119,'LINK'/,' (#)',T12,'(HA)',T24,'(CMS)'T36'(M/M)',
     &      T49'(M)'T60,'(M/S)',T72,'(M/M)',T85,'(M)',T97,'(M)',T109,'(M)',
     &      T119,'(#)',/)
   DO 239 I=1,NSSGRP
          TSAREA=0.0
          DO 238 J=1,3
                 CONST=TBTRYS(I,J)
                 IF(CONST.LE.0) THEN
                         GOTO 238
                 ENDIF
                 TOLINK=0.0
                 DO 231 K=1,NSSGRP
                         IF(CONST.EQ.TRTRYS(K,4)) THEN
                                 TOLINK=1.0
                         ENDIF
231                      CONTINUE
                 A2=AREA(CONST)+ASS(CONST)
                 TSAREA=TSAREA+A2
*
*                CALCULATE THE GENERATED STORM SEWER FLOW
*                USING AN EQUATION DEVELOPED FROM DATA
*                OBTAINED FROM The City of Regina.
*
                 QSS=(0.62*(A2/0.4047)**0.75)*(0.3048**3)
                 E1=ELEV(NODE1(CONST))
                 E2=ELEV(NODE2(CONST))
                 GSLOPE=(E1-E2)/LLINK(CONST)
                 IF(GSLOPE.LE.0) THEN
                         SLOPE=0.000169
                 ELSE
                         SLOPE=GSLOPE
                 ENDIF
                 SSLOPE=SLOPE
                 D=(3.208*QSS*FRICTN/SQRT(SSLOPE))**0.375
                 DD=D
*
*                CHOOSE A PIPE DIAMETER LARGER THAN THE
*                CALCULATED DIAMETER
*
233                      IF(DD.LT.0.300) THEN
                         D3=0.300
                         INDX3=1
                 ELSEIF (DD.LT.0.375) THEN
                         D3=0.375
                         INDX3=2
                 ELSEIF (DD.LT.0.450) THEN
                         D3=0.450
                         INDX3=3
                 ELSEIF (DD.LT.0.525) THEN
                         D3=0.525
                         INDX3=4
```

```
                ELSEIF (DD.LT.0.600) THEN
                        D3=0.600
                        INDX3=5
                ELSEIF (DD.LT.0.750) THEN
                        D3=0.750
                        INDX3=6
                ELSEIF (DD.LT.0.900) THEN
                        D3=0.900
                        INDX3=7
                ELSEIF (DD.LT.1.050) THEN
                        D3=1.050
                        INDX3=8
                ELSEIF (DD.LT.1.200) THEN
                        D3=1.200
                        INDX3=9
                ELSEIF (DD.LT.1.350) THEN
                        D3=1.350
                        INDX3=10
                ELSEIF (DD.LT.1.500) THEN
                        D3=1.500
                        INDX3=11
                ELSEIF (DD.LT.1.650) THEN
                        D3=1.650
                        INDX3=12
                ELSE
                        D3=1.800
                        INDX3=13
                ENDIF
                DIASS(CONST)=D3
                CALL DIAMTR(DIASS(CONST),QSS,SSLOPE,FRICTN,SSVMIN
     &                  ,SSVMAX,VEL)
                IF((VEL.GT.SSVMAX).AND.(D3.LT.1.800)) THEN
                        DD=D3
                        SSLOPE=SLOPE
                        GOTO 233
                ENDIF
                SSDPTH=SSCOVR+DIASS(CONST)+0.100
                SSE1=ELEV(NODE1(CONST))-SSDPTH
                IF(TOLINK.GT.0.5) THEN
                        GOTO 235
                ENDIF
234                     SSE2=SSE1-LLINK(CONST)*SSLOPE
                IF(SSELEV(NODE2(CONST)).GT.SSE2) THEN
                        SSELEV(NODE2(CONST))=SSE2
                ENDIF
                CALL DEPTH (ELEV(NODE1(CONST)),ELEV(NODE2(CONST)),
     &                  SSE1,SSE2,
     &                  DPTH1,DPTH2,ADPTH,DINDX3)
                IF((DPTH2*1.001).LT.SSDPTH) THEN
                        SSE1=SSE1-SSDPTH+DPTH2
                        GOTO 234
                ENDIF
                GOTO 236
235                     IF(SSELEV(NODE1(CONST)).GT.SSE1) THEN
                        SSELEV(NODE1(CONST))=SSE1
                ENDIF
                SSE2=SSELEV(NODE1(CONST))-LLINK(CONST)*SSLOPE
```

```
                    IF(SSELEV(NODE2(CONST)).GT.SSE2) THEN
                            SSELEV(NODE2(CONST))=SSE2
                    ENDIF
                    CALL DEPTH (ELEV(NODE1(CONST)),ELEV(NODE2(CONST)),
     &                      SSELEV(NODE1(CONST)),SSE2,
     &                      DPTH1,DPTH2,ADPTH,DINDX3)
                    IF((DPTH2*1.001).LT.SSDPTH) THEN
                            SSELEV(NODE1(CONST))=SSELEV(NODE1(CONST))-
     &                              SSDPTH+DPTH2
                            GOTO 235
                    ENDIF
236                         CSS(CONST)=COST13(DINDX3,INDX3)
                    SSCOST=SSCOST+CSS(CONST)*LLINK(CONST)
                    DIASS(CONST)=DIASS(CONST)*1000.0
                    WRITE(WUNIT1,237)CONST,A2,QSS,GSLOPE,D,VEL,SSLOPE,DPTH1,
     &                      DPTH2,ADPTH,CONST

*
*
*  save the depth of excavation

            CALL SSSDPTH(CONST,DPTH1,DPTH2)


237                         FORMAT(I4,3X,F10.5,8(2X,F10.5)4X,I4)
238                 CONTINUE
            ASS(TBTRYS(I,4))=ASS(TBTRYS(I,4))+TSAREA
239         CONTINUE
   TLCOST=TLCOST+SSCOST
   CALL ASPLIT(3,SSCOST)
   RETURN
   END
```

B.2.33 SUBROUTINE: UTILITY

```
      SUBROUTINE UTILITY
*****************************************************************
*                                                               *
*                      UTILITY REQUIREMENT                      *
*               (i.e. telephone, gas, and power)                *
*                                                               *
*****************************************************************
*
*  TMULTI - total number of multifamily units
*  TSNGLE - total number of single units
*  TMCOST - total cost of utilities for multifamily units
*  TSCOST - total cost of utilities for single units
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  WUNIT1,WUNIT2,RUNIT1,RUNIT2,ERRFLG

      REAL
     &  MULTI

      COMMON
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA16/COST19,COST20,COST21,COST22,COST23,COST24
     &      /AREA19/TLCOST
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA37/SRCOST(LINNUM),FRONT(LINNUM),ASCHOL,AMULT,ACOM,AOTHER,
     &      ASUB,ASIDE(LINNUM),ASING,TSNGLE,TMULTI,APAVE,AWALK
     &      /AREA44/PWSING(LINNUM),PWMULT(LINNUM),GSSING(LINNUM)
     &      ,GSMULT(LINNUM),PHSING(LINNUM),PHMULT(LINNUM),UTCOST(LINNUM)
*
      TMULTI=0.0
      TSNGLE=0.0

      DO 720 L=1,NLINK
             TMULTI=TMULTI+MULTI(L)
             TSNGLE=TSNGLE+UNITS(L)
             PWSING(L)=UNITS(L)*COST19
             GSSING(L)=UNITS(L)*COST21
             PHSING(L)=UNITS(L)*COST23
             PWMULT(L)=MULTI(L)*COST20
             GSMULT(L)=MULTI(L)*COST22
             PHMULT(L)=MULTI(L)*COST24
             UTCOST(L)=PWSING(L)+GSSING(L)+PHSING(L)+
     &               PWMULT(L)+GSMULT(L)+PHMULT(L)
720          CONTINUE
*
      TMCOST=TMULTI*(COST20+COST22+COST24)
      TSCOST=TSNGLE*(COST19+COST21+COST23)
      TLCOST=TLCOST+TMCOST+TSCOST
*
```

```
*  WRITE(WUNIT1,730)TSNGLE,TSCOST,TMULTI,TMCOST
730        FORMAT(10X,'UTILITIES (# SINGLE FAMILY)',T41,F7.1,T54,F7.0,/,
     &  10X,'            (# MULTI FAMILY)',T41,F7.1,T54,F7.0,/)
*
*
*
   RETURN
   END
```

B.2.34 SUBROUTINE: WATER

```
      SUBROUTINE WATER(WMCOST)
*****************************************************************
*                                                               *
*                        SUBROUTINE WATER                       *
*                                                               *
*     This subroutine designs the WATERMAIN system. For each link,  *
*     the subroutine: (1) estimates the domestic and fire fighting  *
*     water requirement, (2) calculates the pipe size required to   *
*     carry the estimated water flow, and (3) selects a standard    *
*     sized watermain diameter based on the calculated diameter.    *
*                                                               *
*****************************************************************
*  USE      - EXPECTED DOMESTIC WATER USE FOR WATERMAIN
*             CALCULATION (GPM)
*  FLOAD    - SAFETY FACTOR FOR WATER CONSUMPTION REQUIREMENTS
*  CMMCMS   - CUBIC METERS PER MINUTE TO CUBIC METERS PER SECOND CONVERSION
*             FACTOR
*  WMCOST   - TOTAL COST OF THE WATERMAIN ($)
*  TPOP     - TOTAL POPULATION FOR EACH DESTINATION LINK
*  CONST    - TEMPORARY CONSTANT
*  P1       - THE TOTAL POPULATION FOR EACH LINK
*  QDMSTC   - TOTAL DOMESTIC WATER FLOW REQUIREMENT (CU. METERS/MIN.)
*  QFIRE    - TOTAL WATER FLOW REQUIREMENT FOR FIRE FIGHTING
*             PURPOSES (CU. METERS/MIN.)
*  QWM      - TOTAL WATER FLOW REQUIREMENT FOR THE WATERMAIN (CMS)
*  D1       - DIAMETER OF THE WATERMAIN (M)
*  DIAWM    - DIAMETER OF THE WATERMAIN (mm)
*  CWM      - THE PER UNIT COST OF THE WATERMAIN FOR EACH LINK ($)
*  INDX1    - DIAMETER INDEX FOR WATERMAIN (1 - 6)
*
*

      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10,
     &  CMMCMS=1./60

      INTEGER TRTRYW,CONST,WUNIT1,WUNIT2,RUNIT1,RUNIT2,ERRFLG
      REAL LLINK,MULTI
      COMMON
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA6/NWMLNK,WLINKS(LINNUM),NWMGRP,TBTRYW(LINNUM,4)
     &      /AREA9/IQ,L,P(LINNUM),PDS(LINNUM),ASS(LINNUM)
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
     &  /AREA20/ERRFLG
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA22/PEOPLE,WVMIN,FRICTN,USE,FLOAD
      WMCOST=0.0
      WRITE(WUNIT1,200)
200         FORMAT('1WATERMAIN',/X,9('='),//,T11,'TRIBUTARY',T37,
     &      'REQUIRED',/,' LINK',T10,'POPULATION    FLOW RATE    DIAMETER'
     &      ,/,' (#)',T13,'(#)',T26,'(CMS)',T39,'(M)'/)
```

```
      DO 214 I=1,NWMGRP
             TPOP=0.0
             DO 210 J=1,3
                     CONST=TBTRYW(I,J)
                     IF(CONST.LE.0) THEN
                             GOTO210
                     ENDIF
                     P1=(UNITS(CONST)+MULTI(CONST))*PEOPLE+P(CONST)
                     TPOP=TPOP+P1
*
*                 CALCULATE THE DOMESTIC AND FIRE FIGHTING WATER
*                 REQUIREMENT AS DESCRIBED IN:
*                 Standard Handbook of Engineering Calculations
*                 New York, 1972
*                 by Tyler Hicks
*
                     QDMSTC=USE*P1*FLOAD/(24*60)
                     QFIRE=1020*0.003785*SQRT(P1/1000)*(1-0.01*SQRT(P1/1000))
*            NOTE: 0.003785 IS A METRIC CONVERSION FACTOR
                     QWM=(QDMSTC+QFIRE)*CMMCMS
                     D1=SQRT((QWM/WVMIN)*(4.0/3.14159))
                     WRITE(WUNIT1,205)CONST,P1,QWM,D1
205                          FORMAT(I4,T11,F7.0,T25,F7.5,T37,F7.5)
*
*                 CHOOSE A PIPE DIAMETER THE SAME SIZE OR LARGER
*                 THAN THE CALCULATED DIAMETER REQUIRED
*
                     IF(D1.LE.0.150) THEN
                             D1=0.150
                             INDX1=1
                     ELSEIF (D1.LE.0.200) THEN
                             D1=0.200
                             INDX1=2
                     ELSEIF (D1.LE.0.250) THEN
                             D1=0.250
                             INDX1=3
                     ELSEIF (D1.LE.0.300) THEN
                             D1=0.300
                             INDX1=4
                     ELSEIF (D1.LE.0.400) THEN
                             D1=0.400
                             INDX1=5
                     ELSE
                             D1=9.999
                             INDX1=6
                     ENDIF
                     DIAWM(CONST)=D1*1000
                     CWM(CONST)=COST11(INDX1)
                     WMCOST=WMCOST+CWM(CONST)*LLINK(CONST)
210                  CONTINUE
             P(TBTRYW(I,4))=P(TBTRYW(I,4))+TPOP
214          CONTINUE
      CALL ASPLIT(1,WMCOST)
      RETURN
      END
```

B.3 SUBROUTINES CALLED BY OTHER SUBROUTINES

B.3.1 SUBROUTINE: ACCOUT

```
      SUBROUTINE ACCOUT(TYPE,TYCOST)

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &       EVEN,TYPE,TIME,WUNIT1

      REAL
        &       INTRST

      CHARACTER*12 ACCONT

      COMMON
        &       /AREA1/NINT,INTRST(INTNUM),LIFE(INTNUM)
        &       /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &       /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &       ,MNPACC(25,10),OPPACC(25,10)
        &       /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &       ,DEFINE(100),NACCNT
*
*
*
      TIME=3
*
      TANNUL=0.0
      DO 100 J=1,NACONT(TYPE)
              AWORTH=UNFORM(INTRST(NINT),LIFE(NINT),SDCOST(TYPE,J))
              FWORTH=UNFORM2(INTRST(NINT),LIFE(NINT),SDCOST(TYPE,J))
              TANNUL=TANNUL+AWORTH
              WRITE (WUNIT1,500) ACCONT(TYPE,J),SDCOST(TYPE,J),AWORTH
100           CONTINUE
      WRITE (WUNIT1,510) TYCOST,TANNUL
      DO 110 I=1,TIME
              WRITE (WUNIT1,520) TYCOST,TANNUL
110           CONTINUE
      WRITE (WUNIT1,530)

500           FORMAT (34X,A12,2(7X,'$',2X,F10.2))
510           FORMAT (19X,'T O T A L',18X,2(7X,'$',2X,F10.2))
520           FORMAT ('+',45X,2(7X,'$',2X,F10.2))
530           FORMAT (/)
*
*
      END
```

B.3.2 SUBROUTINE: ACCOUT1

```
      SUBROUTINE ACCOUT1(TYPE)

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &      EVEN,TYPE,TIME,WUNIT1

      REAL IMCOST,MGRAD,MGRATE,MNCOST,MINST,MNPACC,MCAPAF

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &      ,MNPACC(25,10),OPPACC(25,10)
        &      /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &      ,DEFINE(100)
        &      /AREA49/IMCOST(25),MGRAD(25),MGRATE(25),MINST(25),MLIFE(25)
        &      ,MNCOST(25,3),MCAPAF(25,10,3)

*
*
*
      TIME=3
*
      DO 100 J=1,NACONT(TYPE)
                WRITE (WUNIT1,500) ACCONT(TYPE,J),(MCAPAF(TYPE,J,K),K=1,2)
100           CONTINUE


500           FORMAT (54X,A12,2(7X,'$',2X,F10.2))
*
*
      END
```

B.3.3 SUBROUTINE: ACCOUT2

```
      SUBROUTINE ACCOUT2(TYPE)

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &      EVEN,TYPE,TIME,OLIFE,WUNIT1

      REAL IOCOST

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
        &      ,MNPACC(25,10),OPPACC(25,10)
        &      /AREA47/ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
        &      ,DEFINE(100)
        &      /AREA50/IOCOST(25),OGRAD(25),OGRATE(25),OINST(25),OLIFE(25)
        &      ,OPCOST(25,3),OCAPAF(25,10,3)

*
*
*
      TIME=3
*
      DO 100 J=1,NACONT(TYPE)
                WRITE (WUNIT1,500) ACCONT(TYPE,J),(OCAPAF(TYPE,J,K),K=1,2)
100           CONTINUE


500           FORMAT (54X,A12,2(7X,'$',2X,F10.2))
*
*
      END
```

B.3.4 SUBROUTINE: ADJ

```
      SUBROUTINE ADJ(CNODE,CLINK,CWD)

*****************************************************************
*                                                               *
*                              ADJ                              *
*                                                               *
*****************************************************************
*
*
*
*  CNODE  - current node in the memory.
*  ATTLIN - the links associate with the current node. ( NOTE: max. link ca
*           attach to the current node is 4 )
*  CLINK  - current link in the memory.
*  CWD    - width of the corresponding road.
*  NL     - number of links attached to the current node.
*  FLAG   - indicate the road width of the corresponding link
*  J      - loop counter
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,CLINK,CNODE,ATTLIN,FLAG
     &      ,EXIT

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM),RCURB
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
*
*
*
*  clear the variable to zero
      DO 100 J=1,4
             ATTLIN(CNODE,J)=0
100          CONTINUE
*
*  set the number of links counter to zero
      NL=0
*
*
*
*  determine the links attach to the current node
      DO 110 J=1,NLINK
             IF ( NODE1(J).EQ.CNODE .OR. NODE2(J).EQ.CNODE ) THEN
                     NL=NL+1
                     ATTLIN(CNODE,NL)=J
```

```
            ENDIF
110         CONTINUE
*
*
*
*  if CLINK is not equal -1, then this routine is called by the
*  PAVEMENT routine, so we must return the corresponding road
*  width of the current link
   IF ( CLINK.EQ.-1 )  GOTO 999
*
*  check for the exit node
*  ( NOTE: exit node always has 1 attached link )
   IF ( NL.EQ.1 )  EXIT(CNODE)=1
*
*  set up the default value of the road width
   CWD=WROAD(CLINK)
   FLAG=0
*
   DO 120 J=1,NL
            L=ATTLIN(CNODE,J)
            IF ( WROAD(L).NE.CWD .AND. WROAD(L)
     &              .NE.0.0 )  THEN
                    FLAG=J
            ENDIF
120         CONTINUE
*
*  determine the corresponding road width of the current link
   IF ( FLAG.NE.0 )  CWD=WROAD(ATTLIN(CNODE,FLAG))
*
*
*
999         RETURN
   END
```

B.3.5 SUBROUTINE: ASPLIT

```
      SUBROUTINE ASPLIT(TYPE,TYCOST)

      INTEGER
     &      TYPE,METHOD,ELEM

      CHARACTER*12 ACCONT,ACCNUM
      CHARACTER*50 DEFINE

      COMMON
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA46/NACONT(25),ACCONT(25,10),PRCENT(25,10),SDCOST(25,10)
     &      ,MNPACC(25,10),OPPACC(25,10)
     &      /AREA47/NACCNT,ACCOST(100),AMCOST(100),AOCOST(100),ACCNUM(100)
     &      ,DEFINE(100)

*     determine which method is going to be used
      TPCENT=0.0
      DO 100 I=1,NACONT(TYPE)
             TPCENT=TPCENT+PRCENT(TYPE,I)
100          CONTINUE
      IF ( TPCENT.LE.1 ) THEN
*     using the ratio or percentage
             METHOD=1
      ELSE
*     the given cost are already broke down
             METHOD=2
      ENDIF
*     apply the method
      IF ( METHOD.EQ.1 )  THEN
             DO 110 I=1,NACONT(TYPE)
               SDCOST(TYPE,I)=TYCOST*PRCENT(TYPE,I)
               DO 120 J=1,NACCNT
                 IF ( ACCNUM(J).EQ.ACCONT(TYPE,I) )  THEN
                   ELEM=J
                 ENDIF
120                    CONTINUE
               ACCOST(ELEM)=ACCOST(ELEM)+SDCOST(TYPE,I)
110                  CONTINUE
      ELSE
             DO 130 I=1,NACONT(TYPE)
               DO 140 J=1,NACCNT
                 IF ( ACCNUM(J).EQ.ACCONT(TYPE,I) )  THEN
                   ELEM=J
                 ENDIF
140                    CONTINUE
               ACCOST(ELEM)=ACCOST(ELEM)+PRCENT(TYPE,I)
130                  CONTINUE
      ENDIF
*
*
*
      RETURN
      END
```

B.3.6 SUBROUTINE: CATCALC

```
      SUBROUTINE CATCALC(CNODE,NFLOW)


*****************************************************************
*                                                               *
*                        SUBROUTINE CATCALC                     *
*                                                               *
*****************************************************************
*
*
*
*  ATTLIN - the links associate with the given node. ( NOTE: max. link can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory.
*  WROAD  - width of the road of a given link
*  CBDPTH - depth of the basin lead from the ground level.
*  CBLEAD - length of the catchbasin lead.
*  CLEAD  - min. length of the catchbasin lead.
*  INDEX  - depth index of catch-basin
*  BASIN  - number of catch-basins required for current node
*  FLINK  - flag indicates the drainage link
*  AVEDEP - average storm sewer elev. for a given link
*  CBCOS  - the cost of the catch-basins for current node
*  NFLOW  - number of drainage flow for current node
*  CBC    - temporary cost
*  IND    - temporary depth index value
*  EXL    - the link with 1 extra catch-basin ( 2 )
*  LCBCOS - the cost of the catch-basins for current link
*  LBASIN - number of catch-basins required for current link
*  J      - loop counter
*  SS     - subscript
*  cost data refer to INDAT2 routine
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  ATTLIN,BASIN,INDEX(4),SS,CNODE,CLINK,EXL

      INTEGER
     &  DINDX3,RUNIT1,RUNIT2,WUNIT1,WUNIT2,FLINK

      REAL
     &      LCBCOS

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
```

```
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNU
     &      /AREA19/TLCOST
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA30/CBLEAD(4,2),CBDPTH(4),FLINK(NODNUM,4),
     &      BASIN(NODNUM),CBCOS(NODNUM),LBASIN(LINNUM),LCBCOS(LINNUM)
     &      /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
*
*
*
*  find the depth index of all links associated with the current
*  node ( except easement )
*
   DO 100 J=1,4
          CLINK=ATTLIN(CNODE,J)
          IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                 IND=INT(AVEDEP(CLINK)/0.5-4)
                 IF ( IND.LT.1 )  IND=1
                 IF ( IND.GT.9 )  IND=9
                 INDEX(J)=IND
          ENDIF
100       CONTINUE
*
*
*
*  calculate the cost
   IF ( BASIN(CNODE).EQ.2 )  THEN
*
*  find the link has drainage
          DO 110 J=1,4
                 IF ( FLINK(CNODE,J).EQ.1 )  THEN
                        CBCOS(CNODE)=(COST16+COST17*CBDPTH(J))
     &                         *2+COST18(INDEX(J))
     &                         *(CBLEAD(J,1)+CBLEAD(J,2))
                        CLINK=ATTLIN(CNODE,J)
                        LBASIN(CLINK)=LBASIN(CLINK)+2
                        LCBCOS(CLINK)=LCBCOS(CLINK)+CBCOS(CNODE)
                 ENDIF
110              CONTINUE
*
   ENDIF
*
   IF ( BASIN(CNODE).EQ.3 )  THEN
*
*  every drainage link has a catch-basin
          DO 120 J=1,4
                 IF ( FLINK(CNODE,J).EQ.1 )  THEN
*  use the min. lead length to min. the cost
                        IF ( CBLEAD(J,1).GT.CBLEAD(J,2) )  THEN
                               SS=2
                        ELSE
                               SS=1
                        ENDIF
```

```
                        CBC=(COST16+COST17*CBDPTH(J))
     &                          +COST18(INDEX(J))*CBLEAD(J,SS)
                        CBCOS(CNODE)=CBCOS(CNODE)+CBC
                        CLINK=ATTLIN(CNODE,J)
                        LBASIN(CLINK)=LBASIN(CLINK)+1
                        LCBCOS(CLINK)=LCBCOS(CLINK)+CBC
                ENDIF
120             CONTINUE
*
        IF ( NFLOW.NE.BASIN(CNODE) )  THEN
*  put the last catch-basin to the link has the min. lead length
                EXL=0
                CLEAD=9999999.0
                DO 130 J=1,4
                        IF ( FLINK(CNODE,J).EQ.1 )  THEN
                                IF ( CBLEAD(J,1).GT.CBLEAD(J,2) )  THEN
                                        SS=1
                                ELSE
                                        SS=2
                                ENDIF
                                IF ( CBLEAD(J,SS).LT.CLEAD )  THEN
                                        EXL=J
                                ENDIF
                        ENDIF
130                     CONTINUE
                IF ( CBLEAD(EXL,1).GT.CBLEAD(EXL,2) )  THEN
                        SS=1
                ELSE
                        SS=2
                ENDIF
                CBC=(COST16+COST17*CBDPTH(EXL))+COST18(INDEX(EXL))
     &                  *CBLEAD(EXL,SS)
                CBCOS(CNODE)=CBCOS(CNODE)+CBC
                CLINK=ATTLIN(CNODE,EXL)
                LBASIN(CLINK)=LBASIN(CLINK)+1
                LCBCOS(CLINK)=LCBCOS(CLINK)+CBC
        ENDIF
   ENDIF
*
   IF ( BASIN(CNODE).EQ.4 )  THEN
*
        DO 140 J=1,4
                IF ( FLINK(CNODE,J).EQ.1 )  THEN
                        IF ( CBLEAD(J,1).GT.CBLEAD(J,2) )  THEN
                                SS=2
                        ELSE
                                SS=1
                        ENDIF
                        CBC=(COST16+COST17*CBDPTH(J))
     &                          +COST18(INDEX(J))*CBLEAD(J,SS)
                        CBCOS(CNODE)=CBCOS(CNODE)+CBC
                        CLINK=ATTLIN(CNODE,J)
                        LBASIN(CLINK)=LBASIN(CLINK)+1
                        LCBCOS(CLINK)=LCBCOS(CLINK)+CBC
                ENDIF
140             CONTINUE
*
```

```
*  find the cost for the last catch-basin
*  put the last catch-basin to the link has the min. lead length
                    EXL=0
                    CLEAD=9999999.0
                    DO 150 J=1,4
                            IF ( FLINK(CNODE,J).EQ.1 )  THEN
                                    IF( CBLEAD(J,1).GT.CBLEAD(J,2) )  THEN
                                            SS=1
                                    ELSE
                                            SS=2
                                    ENDIF
                                    IF ( CBLEAD(J,SS).LT.CLEAD )  THEN
                                            EXL=J
                                    ENDIF
                            ENDIF
150                         CONTINUE
                    IF ( CBLEAD(EXL,1).GT.CBLEAD(EXL,2) )  THEN
                            SS=1
                    ELSE
                            SS=2
                    ENDIF
                    CBC=(COST16+COST17*CBDPTH(EXL))+COST18(INDEX(EXL))
     &                      *CBLEAD(EXL,SS)
                    CBCOS(CNODE)=CBCOS(CNODE)+CBC
                    CLINK=ATTLIN(CNODE,EXL)
                    LBASIN(CLINK)=LBASIN(CLINK)+1
                    LCBCOS(CLINK)=LCBCOS(CLINK)+CBC
   ENDIF
*
*
*
   RETURN
   END
```

B.3.7 SUBROUTINE: CATCALC2

```
      SUBROUTINE CATCALC2(CLINK,CLEAD,CDPTH)


*****************************************************************
*                                                               *
*                      SUBROUTINE CATCALC2                      *
*                                                               *
*****************************************************************
*
*
*
*  CLINK  - current link in the memory.
*  CDPTH - depth of the basin lead from the ground level.
*  CLEAD - length of the catchbasin lead.
*  INDEX  - depth index of catch-basin
*  AVEDEP - average storm sewer elev. for a given link
*  LCBCOS  - the cost of the catch-basins for current link
*  IND    - temporary depth index value
*  cost data refer to INDAT2 routine
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  CLINK

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2

      REAL
     &      LCBCOS,CLEAD(2)

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST16,
     &      COST17,COST18(9)
     &      /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
     &      DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
     &      /AREA19/TLCOST
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA30/CBLEAD(4,2),CBDPTH(4),FLINK(NODNUM,4),
     &      BASIN(NODNUM),CBCOS(NODNUM),LBASIN(LINNUM),LCBCOS(LINNUM)
     &      /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
*
*
*
```

```
*  find the depth index of the current link
*
                    IND=INT(AVEDEP(CLINK)/0.5-4)
                    IF ( IND.LT.1 )  IND=1
                    IF ( IND.GT.9 )  IND=9
                    INDEX=IND
*
*
*
*  calculate the cost
*
                              LCBCOS(CLINK)=(COST16+COST17*CDPTH)
     &                                 *2+COST18(INDEX)*(CLEAD(1)+CLEAD(2)
     &                                 )+LCBCOS(CLINK)
*
*
*
   RETURN
   END
```

B.3.8 SUBROUTINE: CATDEPTH

```
      SUBROUTINE CATDEPTH(CNODE,CLINK,FROM,DSTLEN,CLEAD,CDPTH)


*****************************************************************************
*
*                                    CATDEPTH
*
*****************************************************************************
*
*
*
*  ATTLIN - the links associate with the given node. ( NOTE: max. links can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  PNODE  - previous node in the memory.
*  CNODE  - current node in the memory.
*  SSLINK - storm sewer elev. of a given link.
*  NODE1  - origin node of the current link.
*  NODE2  - dest. node of the current link.
*  SSELE1 - storm sewer elev. of the current link at the current node.
*  SSELE2 - storm sewer elev. of the current link at the previous node.
*  CDPTH  - depth of the basin lead from the ground level.
*  CLEAD  - length of the catchbasin lead.
*  ELEV1  - the elevation difference at current node.
*  ELEV2  - the elevation difference at previous node.
*  ANG    - the slope angle of the pipe.
*  CWD    - corresponding road width of the current link
*  BASHGT - the height between the lead and the bedding.
*  BASBED - the height of the bedding.
*  OFFSET - the distance from the center of the street.
*  AVEDEP - average depth of the storm sewer of the current link
*  DSTLEN - difference in street length ( for calculating link's catch-
*           basins only )
*  DIST1   - the distance from the current node to the catch-basin
*  DIST2   - the distance from the current node to the catch-basin
*  FROM   - identify the call from which part of the main catch-basin routi
*  NEG    - negative flag.
*  J      - loop counter
*  SS,MM  - subscript
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  CNODE,PNODE,CLINK,ATTLIN,SS,FROM

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2

      REAL
     &     CLEAD(2)
```

```
      COMMON
        &       /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
        &       ANGLE(NODNUM)
        &       /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
        &       UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
        &       /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
        &       /AREA18/RK4(2),DSELEV(NODNUM),SSELEV(NODNUM),DIAWM(LINNUM),
        &       DIADS(LINNUM),DIASS(LINNUM),CWM(LINNUM),CDS(LINNUM),CSS(LINNUM)
        &       /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &       /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
        &       /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
        &       /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
        &       /AREA33/BASHGT,BASBED,OFFSET
*
*
*     set the negative flag to 1
      NEG=1
*
*
*
*     determine the previous node of the link and find the
*     corresponding roadwidth
      IF ( NODE1(CLINK).EQ.CNODE )  THEN
              MM=1
              PNODE=NODE2(CLINK)
      ELSE
              MM=2
              PNODE=NODE1(CLINK)
      ENDIF
      CWD=CORWD(CLINK,MM)
*
*
*
*     identify the call from which part of the main catch-basin routine
*     when FROM = 0 , it is calling by the node part of the main catch-basin
*     routine
*     when FROM = 1 , it is calling by the link part of the main catch-basin
*     routine
      IF ( FROM.EQ.0 )  THEN
              DIST1=0.5*CWD
              DIST2=LLINK(CLINK)-0.5*CWD
      ELSE
              DIST1=LLINK(CLINK)-DSTLEN
              DIST2=DSTLEN
      ENDIF
*
*
*
*     determine the slope angle of the pipe.
*
*     find the storm sewer elev. at current node
      DO 100 J=1,4
              IF ( ATTLIN(CNODE,J).EQ.CLINK )  THEN
                      SS=J
              ENDIF
100           CONTINUE
```

```
      SSELE1=SSLINK(CNODE,SS)
*
*     find the storm sewer elev. at previous node
      DO 110 J=1,4
              IF ( ATTLIN(PNODE,J).EQ.CLINK )  THEN
                      SS=J
              ENDIF
110           CONTINUE
      SSELE2=SSLINK(PNODE,SS)
*
      IF ( SSELE1.GT.9000.0 .OR. SSELE2.GT.9000.0 )
     &        THEN
              CLEAD(1)=9999999.99
              CLEAD(2)=9999999.99
              CDPTH=0.0
      ELSE
              ELEV1=ELEV(CNODE)-SSELE1
              ELEV2=ELEV(PNODE)-SSELE2
              AVEDEP(CLINK)=(ELEV1+ELEV2)/2.0
*
*     calculate the lead length and the depth
              IF ( ELEV1.LT.ELEV2 )  THEN
                      ANG=ATAN(ELEV2-ELEV1)/LLINK(CLINK)
                      DPTH=TAN(ANG)*(DIST1)+ELEV1
                      DO 120 J=1,2
                              NEG=-NEG
                              CLEAD(J)=((0.5*WROAD(CLINK)+OFFSET*NEG)
     &                                  **2.0+BASHGT**2.0)**0.5
120                           CONTINUE
                      CDPTH=DPTH+BASBED
              ELSE
                      ANG=ATAN(ELEV1-ELEV2)/LLINK(CLINK)
                      DPTH=TAN(ANG)*(DIST2)
     &                        +ELEV2
                      DO 130 J=1,2
                              NEG=-NEG
                              CLEAD(J)=((0.5*WROAD(CLINK)+OFFSET*NEG)
     &                                  **2.0+BASHGT**2.0)**0.5
130                           CONTINUE
                      CDPTH=DPTH+BASBED
              ENDIF
*
      ENDIF
*
*
*
      RETURN
      END
```

B.3.9 SUBROUTINE: CATLIN

```
      SUBROUTINE CATLIN(ORIGIN,DEST,BRANCH,START,END,FROM,LENGTH)


*****************************************************************
*                                                               *
*                    CATCHBASIN PER STREETLENGTH                *
*                                                               *
*****************************************************************
*
*
*
*  ATTLIN - the links associate with the given node. ( NOTE: max. links can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  ORIGIN - the origin node of the sequence of nodes comprising a street
*           ( one street may be composed of several links )
*  DEST   - the dest node of the sequence of nodes comprising a street
*           ( one street may be composed of several links )
*  BRANCH - the branch of the path
*  STRLEN - street length between two intersections or intersection/dead
*           end/cul-de-sac
*  SPACAT - spacing of catch-basin
*  NBASIN - number of catch-basins per street length ( city block )
*  STRCAT - number of catch-basins per street length ( city block )
*           ( record the values in the memory )
*  PATH   - the links between two intersections or intersection/dead
*           end/cul-de-sac
*  STLEN1 - the total length by adding the spacing of catch-basin
*  STLEN2 - the total length by adding the road length of the link
*  DSTLEN - difference in street length ( for calculating link's catch-
*           basins only, the difference of STLEN2 and STLEN1)
*  SLEN   - street length of an given clink ( value returned from CHECKLEN
*           routine )
*  START  - the starting link
*  END    - the last link
*  FROM   - flag indicates straight street
*  STLENG - length of an section of a street or between two intersections o
*           end/cul-de-sac
*  LENGTH - length of an section of a street
*  JOINT  - the node connects the two given links
*  K,L    - counter
*
*
*
      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &  CLINK,PATH,FROM,START,END,ORIGIN,DEST,BRANCH

      INTEGER
        &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,STRCAT,ATTLIN

      REAL
```

```
     &  CLEAD(2),LENGTH

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA25/SPAMAN,SPALIT,SPACAT
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
     &      /AREA30/CBLEAD(4,2),CBDPTH(4),FLINK(NODNUM,4),
     &      BASIN(NODNUM),CBCOS(NODNUM),LBASIN(LINNUM),LCBCOS(LINNUM)
     &      ,STRCAT(NODNUM,NODNUM,4)
*
*
*
*  identify the street condition/oriention
*  when FROM = 0, this is a straight street ( no corner )
      IF ( FROM.EQ.0 )  THEN
            STLENG=STRLEN(ORIGIN,DEST,BRANCH)
      ELSE
            STLENG=LENGTH
      ENDIF
*
*  determine the number of catch-basins per street length ( or city block )
*  or section of the street ( intersection/corner )
      NBASIN=(INT(STLENG/SPACAT-0.5))*2
      STRCAT(ORIGIN,DEST,BRANCH)=NBASIN+STRCAT(ORIGIN,DEST,BRANCH)
*
*  determine which link has the catchbasins
*  the total length by adding the spacing of catch-basin
      STLEN1=0.0
*  the total length by adding the road length of the link
      STLEN2=0.0
*
*  set up the start counter
      K=START-1
      L=0
*
      STLEN1=STLEN1+SPACAT
100         IF ( STLEN1.LE.STLEN2 )  THEN
            CLINK=PATH(ORIGIN,BRANCH,K)
            IF ( NBASIN.GT.0 )  THEN
              LBASIN(CLINK)=LBASIN(CLINK)+2
              NBASIN=NBASIN-2
              DSTLEN=STLEN2-STLEN1
*  find the connecting node between two given links
              IF ( L.GE.2 )  THEN
                CNODE=NODE1(CLINK)
                DO 110 I=1,2
                  IF ( CNODE.EQ.NODE1(CLINK) )  THEN
                    CNODE=NODE2(CLINK)
                  ELSE
                    CNODE=NODE1(CLINK)
                  ENDIF
                  DO 120 J=1,4
```

```
                    IF ( ATTLIN(CNODE,J).EQ.0 )  GOTO 120
                    IF ( ATTLIN(CNODE,J).EQ.PATH(ORIGIN,BRANCH,K-1) )  THEN
                      JOINT=CNODE
                    ENDIF
120                      CONTINUE
110                    CONTINUE
              ELSE
                IF ( FROM.EQ.0 )  THEN
                  JOINT=ORIGIN
                ELSE
                  IF ( NTYPE(NODE1(CLINK)).EQ.2 )  THEN
                    JOINT=NODE1(CLINK)
                  ELSE
                    JOINT=NODE2(CLINK)
                  ENDIF
                ENDIF
              ENDIF
*
              CALL CATDEPTH(JOINT,CLINK,1,DSTLEN,CLEAD,CDPTH)
              CALL CATCALC2 (CLINK,CLEAD,CDPTH)
              STLEN1=STLEN1+SPACAT
              GOTO 100
            ENDIF
    ELSE
            L=L+1
            K=K+1
            IF ( K.LE.END)  THEN
              CLINK=PATH(ORIGIN,BRANCH,K)
              CALL CHECKLEN(CLINK,SLEN)
              STLEN2=STLEN2+SLEN
              GOTO 100
            ENDIF
    ENDIF
*
*
*
    RETURN
    END
```

B.3.10 SUBROUTINE: CATNODE

```
      SUBROUTINE CATNODE(CNODE)


*****************************************************************
*                                                               *
*                    CATCHBASIN AT NODE TYPE 1 OR 2             *
*                                                               *
*****************************************************************
*
*
*
*  NTYPE  - node type.
*  ATTLIN - the links associate with the given node. ( NOTE: max. links can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory.
*  CDPTH  - depth of the basin lead from the ground level. ( value returned
*           from CATDEPTH routine )
*  CLEAD  - length of the catchbasin lead. ( value returned from
*           CATDEPTH routine )
*  WROAD  - width of the road of a given link.
*  BASIN  - number of catch-basins required for current node.
*  CBCOS  - the cost of the catch-basins for current node.
*  SSLINK - storm sewer elev. of a given link.
*  SSELE1 - storm sewer elev. of a type 2 node
*  BASHGT - the height between the lead and the bedding.
*  BASBED - the height of the bedding.
*  LBASIN - number of catch-basins required for current link ( include the
*           the catch-basins for associated node )
*  LCBCOS - the cost of the catch-basins for current link ( include the
*           the catch-basins for associated node )
*  NL     - number of street links associsted with a given node
*  INDEX  - depth index of catch-basin
*  IND    - temporary depth index value
*  J      - loop counter
*  SS     - subscript
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  CLINK,CNODE,ATTLIN,BASIN,SS,CUL

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,LINASS(4)

      REAL
     &  LCBCOS

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
```

```
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA15/COST11(6),COST12(9,5),COST13(9,14),COST14,COST15,COST
     &      COST17,COST18(9)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA30/CBLEAD(4,2),CBDPTH(4),FLINK(NODNUM,4),
     &      BASIN(NODNUM),CBCOS(NODNUM),LBASIN(LINNUM),LCBCOS(LINNUM)
     &      ,STRCAT(NODNUM,NODNUM,4)
     &      /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
     &      /AREA33/BASHGT,BASBED,OFFSET
*
*
*
*  add one catch-basin to the cul-de-sac/dead end/corner of the street
*  and find the cost of the catch-basin
*
   IF ( NTYPE(CNODE).EQ.1 )  THEN
*  find out the link associated with this type 1 node
          DO 100 J=1,4
            CLINK=ATTLIN(CNODE,J)
            IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
              SS=J
            ENDIF
100               CONTINUE
   ELSE
*  for type 2 node, find the link has the highest storm sewer elev
          SSELE1=0.0
          DO 110 J=1,4
            IF ( SSLINK(CNODE,J).GT.SSELE1 .AND. SSLINK(CNODE,J).LT.
     &                9000.0 )  THEN
              SS=J
            ENDIF
110               CONTINUE
   ENDIF
*
   CLINK=ATTLIN(CNODE,SS)
*
*  determine the length of the catch-basin lead and depth
   CDPTH=ELEV(CNODE)-SSLINK(CNODE,SS)+BASBED
   CLEAD=((0.5*WROAD(CLINK))**2.0+BASHGT**2.0)**0.5
*
*  find the depth index
   IND=INT((ELEV(CNODE)-SSLINK(CNODE,SS))/0.5-4)
   IF ( IND.LT.1 )  IND=1
   IF ( IND.GT.9 )  IND=9
   INDEX=IND
*
*  calculate the cost to install the catch-basin
   CBCOS(CNODE)=(COST16+COST17*CDPTH)+COST18(INDEX)*CLEAD
   BASIN(CNODE)=1
*  for type 2 node, the cost go to the link with the highest storm sewer
*  elevation
   IF ( NTYPE(CNODE).EQ.2 )  THEN
          LCBCOS(CLINK)=CBCOS(CNODE)+LCBCOS(CLINK)
          LBASIN(CLINK)=LBASIN(CLINK)+1
   ELSE
```

```
            IF ( NTYPE(NODE1(CLINK)).EQ.2 .OR. NTYPE(NODE2(CLINK))
     &              .EQ.2 )  THEN
              CALL CULCHK(CLINK,CUL,M1,M2,LINK1,A)
              IF ( CUL.EQ.0 )  THEN
                LCBCOS(CLINK)=CBCOS(CNODE)+LCBCOS(CLINK)
                LBASIN(CLINK)=LBASIN(CLINK)+1
              ELSE
                LCBCOS(LINK1)=CBCOS(CNODE)+LCBCOS(LINK1)
                LBASIN(LINK1)=LBASIN(LINK1)+1
              ENDIF
            ELSE
              LCBCOS(CLINK)=CBCOS(CNODE)+LCBCOS(CLINK)
              LBASIN(CLINK)=LBASIN(CLINK)+1
            ENDIF
    ENDIF
*
*
*
    RETURN
    END
```

B.3.11 SUBROUTINE: CHECKLEN

```
      SUBROUTINE CHECKLEN(CLINK,SLEN)

*****************************************************************
*                                                               *
*                       CHECK STREET LENGTH                     *
*                                                               *
*****************************************************************
*
*
*
*  NTYPE  - node type.
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory.
*  PNODE  - previous node in the memory
*  NODE1  - origin node of the current link.
*  NODE2  - dest. node of the current link.
*  LLINK  - length of a given link.
*  LROAD  - road length of a given link.
*  RADIUS - radius of the cul-de-sac/tube sac
*  SLEN   - street length of an given clink
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,CLINK,PNODE,CNODE

      REAL
     &  LLINK,LROAD

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      RCURB
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
*
*
*
*  find the correct street length
*  since, cul-de-sac/tube sac has different road length and street length,
*  so we must find the street length for cul-de-sac/tube sac separately
*  current node always refers to the node with radius ( type 1 )
      IF ( NTYPE(NODE1(CLINK)).EQ.1 .OR. NTYPE(NODE2(CLINK)).EQ.1 )  THEN
              IF ( NTYPE(NODE1(CLINK)).EQ.1 )  THEN
                CNODE=NODE1(CLINK)
                PNODE=NODE2(CLINK)
              ELSE
                CNODE=NODE2(CLINK)
                PNODE=NODE1(CLINK)
```

```
            ENDIF
            IF ( RADIUS(CNODE).NE.0.0 )  THEN
              IF ( NTYPE(PNODE).NE.2 )  THEN
                SLEN=LLINK(CLINK)-(CORWD(CLINK,1)+CORWD(CLINK,2))/2.0+
     &                  RADIUS(CNODE)
              ELSE
                SLEN=LLINK(CLINK)+RADIUS(CNODE)
              ENDIF
            ELSE
              SLEN=LROAD(CLINK)
            ENDIF
    ELSE
            SLEN=LROAD(CLINK)
    ENDIF
*
*
*
    RETURN
    END
```

B.3.12 SUBROUTINE: CLINE

```
      SUBROUTINE CLINE(XSPA,XSEC,COL,CON1,CON2)

*****************************************************************
*                                                               *
*                      CREATE LINE CHARACTERS                   *
*                                                               *
*****************************************************************
*
*
*
*  XSPA   - number of spaces in each section.
*  XSEC   - number of sections.
*  XTOTAL - spacing counter.
*  MCOL   - max. columns of the table.
*  COL    - starting column.
*  LINE   - line characters.
*  LINE1  - line characters.
*  LINE2  - line characters.
*  LINE3  - line characters.
*  LINE4  - line characters.
*  CON1   - condition flag indicates the shift of a section.
*  CON2   - condition flag indicates the shift of a section.
*  I      - loop counter.
*
*
*
      INTEGER XSPA,XSEC,CON1,CON2,COL,XTOTAL

      CHARACTER*132 LINE,LINE1,LINE2,LINE3,LINE4

      COMMON
     &     /AREA39/LINE,LINE1,LINE2,LINE3,LINE4
*
*
*
*  clear the character variables
      DO 90 I=1,132
              LINE(I:I)=' '
              LINE1(I:I)=' '
              LINE2(I:I)=' '
              LINE3(I:I)=' '
              LINE4(I:I)=' '
90    CONTINUE
*
*
*
      MCOL=XSPA*XSEC
*
*  start to create the line characters
      DO 100 I=2,MCOL+COL-1
              LINE(I:I)='-'
100           CONTINUE
      LINE(1:1)='!'
      LINE(MCOL+COL:MCOL+COL)='!'
```

```
      IF ( COL.EQ.0 )  THEN
              XTOTAL=XSPA
      ELSE
              XTOTAL=COL
      ENDIF
      DO 110 I=2,MCOL+COL-1
              IF ( I.EQ.XTOTAL )  THEN
                LINE1(I:I)='+'
                XTOTAL=XTOTAL+XSPA
              ELSE
                LINE1(I:I)='-'
              ENDIF
110           CONTINUE
      LINE1(1:1)='!'
      LINE1(MCOL+COL:MCOL+COL)='!'
      IF ( CON1.EQ.1 ) THEN
              XTOTAL=COL+XSPA
      ELSE
              IF ( COL.EQ.0 )  THEN
                      XTOTAL=XSPA
              ELSE
                      XTOTAL=COL
              ENDIF
      ENDIF
      DO 120 I=2,MCOL+COL-1
              IF ( I.EQ.XTOTAL )  THEN
                LINE2(I:I)='+'
                XTOTAL=XTOTAL+XSPA*2
              ELSE
                LINE2(I:I)='-'
              ENDIF
120           CONTINUE
      LINE2(1:1)='!'
      IF ( COL.NE.0 )  THEN
              LINE2(COL:COL)='+'
      ENDIF
      LINE2(MCOL+COL:MCOL+COL)='!'
      IF ( CON1.EQ.1 ) THEN
              XTOTAL=COL+XSPA
      ELSE
              IF ( COL.EQ.0 )  THEN
                      XTOTAL=XSPA
              ELSE
                      XTOTAL=COL
              ENDIF
      ENDIF
      DO 130 I=2,MCOL+COL-1
              IF ( I.EQ.XTOTAL )  THEN
                LINE3(I:I)='!'
                LINE4(I:I)='!'
                XTOTAL=XTOTAL+XSPA*2
              ELSE
                LINE3(I:I)=' '
                LINE4(I:I)=' '
              ENDIF
130           CONTINUE
      LINE3(1:1)='!'
```

```
      LINE3(MCOL+COL:MCOL+COL)='!'
      LINE4(1:1)='!'
      LINE4(MCOL+COL:MCOL+COL)='!'
      IF ( COL.NE.0 )  THEN
              LINE3(COL:COL)='!'
              LINE4(COL:COL)='!'
      ENDIF
*
*
*
      RETURN
      END
```

B.3.13 SUBROUTINE: COR

```
      SUBROUTINE COR(CNODE,I,BULB,ASAC,LEN)

      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,CNODE,COLINK,CODE,
        &      BULB,ATTLIN,C

      REAL
        &  LLINK,LROAD,LENGTH,KM,ACIR(2),AEXC(2),H(2),LEN

      COMMON
        &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
        &      ANGLE(NODNUM),RCURB
        &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
        &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
        &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA26/ADJWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
        &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)

      ASAC=0.0
      LEN=0.0

      DO 222 MM=1,2
              ACIR(MM)=0.0
              H(MM)=0.0
              AEXC(MM)=0.0
222           CONTINUE

      PI=ACOS(-1.0)

*
      DO 111 J=1,BULB

      IF ( J.GT.1 )  THEN

              IF ( CNODE.EQ.NODE1(I) )  THEN
                      CNODE=NODE2(I)
              ELSE
                      CNODE=NODE1(I)
              ENDIF

      ENDIF

      RAD=0.5*RADIUS(CNODE)

*  FIND THE AREA FOR A 1/2 CIRCLE ( BECAUSE OF THE SYMMETRY OF THE CURVE)

      ACIR(J)=0.5*PI*RAD**2.0


*  APPLYING HERON'S FORMULA TO FIND THE HEIGHT OF THE TRIANGLE
```

```
      S1=0.5*(WROAD(I)+RADIUS(CNODE))

*     ERROR CHECKING

      IF ( S1.LE.WROAD(I) .OR. S1.LE.RAD )  THEN

              WRITE(5,250)

      ENDIF
      ATRI=(S1*(S1-WROAD(I))*(S1-RAD)*(S1-RAD))**0.5
      H(J)=ATRI/WROAD(I)*2.0

*     FIND THE AREA OF THE SECTOR

      ANG=2.0*ATAN(WROAD(I)/2.0/H(J))
      ASEC=0.5*ANG*RAD**2.0
      AEXC(J)=ASEC-ATRI


111           CONTINUE


*     AREA OF THE STRAIGHT PART OF THE CURVE

      ROAD=LLINK(I)-H(1)-H(2)
      AEND=ROAD*WROAD(I)
      LEN=LLINK(I)


*     AREA OF THE CURB

      ACUB=0.125*PI*CURB**2.0

*     AREA OF THE WHOLE CURVE

      ASAC=AEND+ACIR(1)+ACIR(2)-AEXC(1)-AEXC(2)-BULB*ACUB


250           FORMAT (//,'ERROR FROM THE COR ROUTINE',//)


      RETURN
      END
```

B.3.14 SUBROUTINE: CORCHK

```
      SUBROUTINE CORCHK(ORIGIN,BRANCH,NBL,BULB)


*****************************************************************
*                                                               *
*                          CORNER CHECK                         *
*                                                               *
*****************************************************************
*
*
*
*  NTYPE  - node type.
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory ( always refers to a type 2 node ).
*  PATH   - the links between two intersections or intersection/dead
*           end/cul-de-sac
*  NBL    - number of links between two intersections or intersection/dead
*           end/cul-de-sac
*  ORIGIN - the origin of the trace
*  BRANCH - the branch of the origin node
*  N1     - origin node of a given link
*  N2     - dest. node of a given link
*  BULB   - a flag indicates the bulb curve(s) found.
*  L      - loop counter
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  CLINK,CNODE,PATH,BRANCH,ORIGIN,BULB

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
*
*
*
*  set bulb/corner flag to zero
      BULB=0
*
*  trace all the links between two intersections or intersection/dead
*  end/cul-de-sac and find the bulb/corner
      DO 100 L=1,NBL
*
            CLINK=PATH(ORIGIN,BRANCH,L)
*  find the origin and dest. node
            N1=NODE1(CLINK)
```

```
            N2=NODE2(CLINK)
            IF ( NTYPE(N1).EQ.2 .OR. NTYPE(N2).EQ.2 )  THEN
*  current node always refers to a type 2 node
                    IF ( NTYPE(N1).EQ.2 )  THEN
                            CNODE=N1
                    ELSE
                            CNODE=N2
                    ENDIF
*  check the radius of the node, if the node has a radius,
*  then this is a corner/bulb
                    IF ( RADIUS(CNODE).NE.0.0 )  THEN
                            BULB=BULB+1
                    ENDIF
            ENDIF
100         CONTINUE
*
*
*
   RETURN
   END
```

B.3.15 SUBROUTINE: CORNER

```
      SUBROUTINE CORNER(ORIGIN,DEST,BRANCH,NBL)


*****************************************************************
*                                                               *
*                    CATCHBASIN AT THE CORNER                   *
*                                                               *
*****************************************************************
*
*
*
*  NTYPE  - node type.
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory ( always refers to a type 2 node ).
*  PATH   - the links between two intersections or intersection/dead
*           end/cul-de-sac
*  NBL    - number of links between two intersections or intersection/dead
*           end/cul-de-sac
*  ORIGIN - the origin node of the sequence of nodes comprising a street
*           ( one street may be composed of several links )
*  DEST   - the dest node of the sequence of nodes comprising a street
*           ( one street may be composed of several links )
*  BRANCH - the branch of the path
*  STRCAT - number of catch-basins per street length ( city block )
*           ( record the values in the memory )
*  N1     - origin node of a given link
*  N2     - dest. node of a given link
*  LENGTH - length of an section of a street
*  RLEN   - street length of an given link
*  START  - starting link
*  L      - loop counter
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  CLINK,CNODE,PATH,BRANCH,ORIGIN,DEST,START,STRCAT

      REAL
     &      LENGTH

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
     &      /AREA30/CBLEAD(4,2),CBDPTH(4),FLINK(NODNUM,4),
     &      BASIN(NODNUM),CBCOS(NODNUM),LBASIN(LINNUM),LCBCOS(LINNUM)
     &      ,STRCAT(NODNUM,NODNUM,4)
*
*
```

```
*
*  trace all the links between two intersections or intersection/dead
*  end/cul-de-sac and find the bulb/corner
   LENGTH=0.0
   DO 100 L=1,NBL
*
            IF ( LENGTH.EQ.0 )  THEN START=L
            CLINK=PATH(ORIGIN,BRANCH,L)
*  find the origin and dest. node
            N1=NODE1(CLINK)
            N2=NODE2(CLINK)
            IF ( NTYPE(N1).EQ.2 .OR. NTYPE(N2).EQ.2 )  THEN
*  current node always refers to a type 2 node
              IF ( NTYPE(N1).EQ.2 )  THEN
                CNODE=N1
              ELSE
                CNODE=N2
              ENDIF
*  check the radius of the node, if the node has a radius,
*  then this is a corner/bulb .
              IF ( RADIUS(CNODE).NE.0.0 )  THEN
                CALL CATNODE(CNODE)
                STRCAT(ORIGIN,DEST,BRANCH)=STRCAT(ORIGIN,DEST
     &                  ,BRANCH)+1
                CALL CATLIN(ORIGIN,DEST,BRANCH,START,L,1,LENGTH)
                LENGTH=0.0
              ELSE
                CALL CHECKLEN(CLINK,SLEN)
                LENGTH=LENGTH+SLEN
              ENDIF
            ENDIF
100         CONTINUE
*
*
*
   RETURN
   END
```

B.3.16 SUBROUTINE: COUNTER

```
      SUBROUTINE COUNTER


*****************************************************************
*                                                               *
*            COUNT NUMBER OF 3 WAY/4 WAY/CUL-DE-SAC/EXIT        *
*                                                               *
*****************************************************************
*
*
*
*  NDEAD  - number of dead-end.
*  NCUL   - number of cul-de-sacs.
*  N3WAY  - number of three way intersections.
*  N4WAY  - number of four way intersections.
*  NEXIT  - number of exits.
*  RADIUS - radius of the cul-de-sac/tube sac
*  NTYPE  - node type.
*
*
*
      PARAMETER
     &     NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &     EXIT

      COMMON
     &     /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &     RCURB
     &     /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &  /AREA41/NCUL,N3WAY,N4WAY,NEXIT,NDEAD
*
*
*
*  set the counter to zero
      NCUL=0
      N3WAY=0
      N4WAY=0
      NEXIT=0
      NDEAD=0
*
*
*
*  check the node type, and its radius for calculating number of
*  cul-de-sac/dead end/three way/four way/exit
      DO 100 N=1,NNODE
            IF ( NTYPE(N).EQ.3 .OR. NTYPE(N).EQ.4 )  THEN
              IF ( NTYPE(N).EQ.3 )  THEN
                IF ( EXIT(N).EQ.1 )  THEN
                  NEXIT=NEXIT+1
                ELSE
                  N3WAY=N3WAY+1
                ENDIF
```

```
              ELSE
                N4WAY=N4WAY+1
              ENDIF
            ENDIF
*
            IF ( NTYPE(N).EQ.1 .OR. RADIUS(N).NE.0.0 )  THEN
              NCUL=NCUL+1
            ELSE
              IF ( NTYPE(N).EQ.1 )  THEN
                NDEAD=NDEAD+0
              ENDIF
            ENDIF
100         CONTINUE
*
*
*
   RETURN
   END
```

B.3.17 **SUBROUTINE: CULCHK**

```
      SUBROUTINE CULCHK(I,CUL,N1,N2,LINK1,LEN)


*****************************************************************
*                                                               *
*                          CUL-DE-SAC CHECK                     *
*                                                               *
*****************************************************************
*
*
*
*  NTYPE  - node type.
*  ATTLIN - the links associate with the given node. ( NOTE: max. links can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  PNODE  - previous node in the memory
*  CNODE  - current node in the memory
*  WROAD  - width of the road of a given link.
*  LEN    - total length links
*  LINK1  - the link between type 2 node and others
*  LINK2  - the link between type 1 node and type 2 node
*  NODE1  - origin node of the current link.
*  NODE2  - dest. node of the current link.
*  N2     - type 1 node of the cul-de-sac
*  N1     - other type of node of the cul-de-sac
*  CUL    - cul-de-sac type 2 node flag
*  J      - loop counter
*
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  ATTLIN,RUNIT1,RUNIT2,WUNIT1,WUNIT2,PNODE,CNODE,CLINK,CUL

      REAL
     &  LLINK,LEN

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
*
*
*
*  set cul-de-sac type 2 node flag and total length links to zero
      CUL=0
      LEN=0.0
*
```

```
*
*
   CLINK=I
*  identify the previous node and current node
*  current node always refers to a type 2 node
   IF ( NTYPE(NODE1(CLINK)).EQ.2 )  THEN
           CNODE=NODE1(CLINK)
           PNODE=NODE2(CLINK)
   ELSE
           CNODE=NODE2(CLINK)
           PNODE=NODE1(CLINK)
   ENDIF
*
*
*
*  identify the type of the previous node
*  if the node type is 1,  then save the link between type 1 node
*  and type 2 node as LINK2
*  otherwise, save the link between type 2 node and other node type
*  as LINK1
   LINK1=0
   LINK2=0
   IF ( NTYPE(PNODE).EQ.1 )  THEN
           LINK2=CLINK
           DO 100 J=1,4
             CLINK=ATTLIN(CNODE,J)
             IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
               IF ( CLINK.NE.LINK2 )  THEN
                 LINK1=CLINK
               ENDIF
             ENDIF
100                CONTINUE
   ELSE
           LINK1=CLINK
           DO 110 J=1,4
             CLINK=ATTLIN(CNODE,J)
             IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
               IF ( CLINK.NE.LINK1 )  THEN
                 LINK2=CLINK
               ENDIF
             ENDIF
110                CONTINUE
   ENDIF
*
   IF ( LINK1.NE.0 .AND. LINK2.NE.0 )  THEN
*
*  find the dest. node and origin node
*  dest. node always refers to type 1 node
*
*  origin node
           IF ( NTYPE(NODE1(LINK1)).EQ.2 )  THEN
             N1=NODE2(LINK1)
           ELSE
             N1=NODE1(LINK1)
           ENDIF
*
*  dest. node
```

```
            IF ( NTYPE(NODE1(LINK2)).EQ.2 )  THEN
              N2=NODE2(LINK2)
            ELSE
              N2=NODE1(LINK2)
            ENDIF
*
*
*
*  identify the type of the street
            IF ( RADIUS(N2).NE.0.0 )  THEN
              CUL=1
              LEN=LLINK(LINK1)+LLINK(LINK2)
            ENDIF
*
   ENDIF
*
*
   RETURN
   END
```

B.3.18 SUBROUTINE: DEPTH

```
      SUBROUTINE DEPTH (E1,E2,PE1,PE2,D1,D2,DPTH,DI)
*****************************************************************
*                                                               *
*                    SUBROUTINE DEPTH                           *
*                                                               *
*       THIS SUBROUTINE CALCULATES THE APPROPRIATE DEPTH INDEX  *
*                                                               *
*****************************************************************
*
*  E1       - GROUND ELEVATION OF ORIGIN NODE [NODE1] (M)
*  E2       - GROUND ELEVATION OF DESTINATION NODE [NODE2] (M)
*  PE1      - ELEVATION OF PIPE AT THE ORIGIN NODE (M)
*  PE2      - ELEVATION OF THE PIPE AT THE DESTINATION NODE (M)
*  DI       - DEPTH INDEX (1 - 9)
*
      INTEGER DI,RUNIT1,RUNIT2,WUNIT1,WUNIT2
      COMMON  /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
      D1=E1-PE1
      D2=E2-PE2
      DPTH=(D1+D2)/2.0
      DI=INT(DPTH/0.5-4)
      IF(DI.LT.1) THEN
              DI=1
      ELSEIF (DI.GT.8) THEN
              DI=9
      ENDIF
      RETURN
      END
```

B.3.19 SUBROUTINE: DIAMTR

```
      SUBROUTINE DIAMTR (D,Q,M,N,VMIN,VMAX,V)
*****************************************************************
*                                                               *
*                    SUBROUTINE DIAMTR                          *
*                                                               *
*       THIS SUBROUTINE USES THE DETERMINED DIAMETER TO         *
*       CALCULATE THE SLOPE REQUIRED TO MEET OR EXCEED THE      *
*       MINIMUM EXCEPTABLE VELOCITY                             *
*                                                               *
*****************************************************************
*
*  D        - DIAMETER OF PIPE (M)
*  Q        - REQUIRED FLOW IN PIPE (CMS)
*  M        - SLOPE OF PIPE
*  N        - MANNING'S ROUGHNESS COEFFICIENT
*  VMIN     - MINIMUM EXCEPTABLE WATER VELOCITY WITHIN PIPE (M/S)
*  HDEPTH   - OVER-ESTIMATED VALUE OF DEPTH OF WATER IN PIPE (M)
*  LDEPTH   - UNDER-ESTIMATED VALUE OF DEPTH OF WATER IN PIPE (M)
*  DEPTH    - ESTIMATED VALUE OF DEPTH OF WATER IN PIPE (M)
*             (i.e. DEPTH = AVERAGE OF HDEPTH & LDEPTH)
*  PRCNT    - THE PERCENTAGE THAT THE DEPTH IS OF THE DIAMETER
*  WP       - WETTED PERIMETER (M)
*  AREA     - CROSS SECTIONAL AREA OF WATER IN THE PIPE (SQ.M)
*  R        - HYDRAULIC RAIUS (M)
*  QEST     - AN ESTIMATE OF THE FLOW AT THE ESTIMATED DEPTH
*             USING MANNING'S EQUATION
*  DIFF     - THE DIFFERENCE BETWEEN THE ESTIMATED FLOW AND
*             THE ACTUAL REQUIRED FLOW
*  V        - ACTUAL VELOCITY OF WATER IN PIPE (M/S)
*
*

      REAL M,N,LDEPTH
      INTEGER RUNIT1,RUNIT2,WUNIT1,WUNIT2
      COMMON  /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
*
*   DETERMINE THE DEPTH OF WATER IN THE PIPE BY QUESSING AT THE DEPTH
*   USING THE BISECTION METHOD AND COMPARING [QEST] WITH [Q]
*
      ICOUNT=0
1005        HDEPTH=D
      LDEPTH=0.0
1010        DEPTH=(HDEPTH+LDEPTH)/2.0
      PRCNT=ACOS((D/2.0-DEPTH)/(D/2.0))/3.14159
      WP=PRCNT*3.14159*D
      AREA=PRCNT*3.14159*D*D/4.0-(D/2.0-DEPTH)*
     &      SQRT((D/2.0)**2-(D/2.0-DEPTH)**2)
      R=AREA/WP
      QEST=AREA*R**(2.0/3.0)*SQRT(M)/N
*
*   DETERMINE IF [QEST] AND [Q] ARE APPROXIMATELY EQUAL
*   IF THEY ARE NOT WITHIN DEFINED TOLERANCES, THEN DETERMINE
*   IF [QEST] IS AN OVER-ESTIMATE OR AN UNDER-ESTIMATE
*
      DIFF=(QEST-Q)/((QEST+Q)/2.0)
```

```
      IF((ABS(DIFF)).LE.0.005) THEN
              GOTO1020
      ENDIF
      IF(DIFF.LT.0) THEN
              LDEPTH=DEPTH
      ELSE
              HDEPTH=DEPTH
      ENDIF
*
*     CHECK FOR AN ENDLESS-LOOP.
*     - IF [HDEPTH] AND [LDEPTH] ARE APPROXIMATELY EQUAL, OR
*     - IF [ICOUNT] IS GREATER THAN 999 ITERATIONS
*     THEN EXIT THE ITERATIVE LOOP.
*
      IF(ABS((HDEPTH-LDEPTH)/((HDEPTH+LDEPTH)/2.0)).LT.0.0001) THEN
              GOTO 1030
      ENDIF
      IF(ICOUNT.GT.999) THEN
              GOTO1030
      ELSE
              ICOUNT=ICOUNT+1
      ENDIF
      GOTO 1010
*
*      CHECK THE WATER VELOCITY AND ENSURE THAT [VMIN] AND [VMAX]
*      ARE MET BY ADJUSTING THE SLOPE
*
1020          V=R**(2.0/3.0)*SQRT(M)/N
      IF(V.LT.VMIN) THEN
              M=(VMIN*1.001*N/R**(2.0/3.0))**2
              GOTO 1005
      ELSEIF(V.GT.VMAX) THEN
              M=(VMAX*0.999*N/R**(2.0/3.0))**2
              GOTO 1005
      ENDIF
1030          RETURN
      END
```

B.3.20 SUBROUTINE: DMAKER

```
      SUBROUTINE DMAKER(CNODE,QUAN,LINK)


*****************************************************************************
*
*                              DECISION MAKER
*
*****************************************************************************
*
*
*
*  ATTLIN - the links associate with the current node. ( NOTE: max. link car
*           attach to the current node is 4 )
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory.
*  NTYPE  - node type.
*  TROAD  - type of road/street
*  EXIT   - exit node.
*  NCOL   - number of collector streets
*  QUAN   - number of signs/lights/manholes at the node ( loop controller )
*  LINK   - stores the collector/major local link
*  WROAD  - road width of a given link
*  CUL    - flag indicates cul-de-sac
*  N1     - origin node of the current link.
*  N2     - dest. node of the current link.
*  J,N    - loop counter
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &     CLINK,CNODE,QUAN,LINK(3),ATTLIN,TROAD,EXIT,CUL

      COMMON
     &     /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &     RCURB
     &     /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &     UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &     /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &     /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &     /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
*
*
*
*  set the collector/major local link to zero
      LINK(1)=0
      LINK(2)=0
      LINK(3)=0
*  set collector street counter to zero
      NCOL=0
*
*
```

```
*
*  count number of collector streets
   DO 100 J=1,4
            CLINK=ATTLIN(CNODE,J)
            IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
              IF ( TROAD(CLINK).EQ.1 )  THEN
                NCOL=NCOL+1
              ENDIF
            ENDIF
100         CONTINUE
*
*
*
*  determine the collector/major local link
   IF ( NCOL.GE.1 )  THEN
            IF ( NCOL.GE.2 )  THEN
              DO 110 N=1,QUAN
                DO 120 J=1,4
                  CLINK=ATTLIN(CNODE,J)
                  IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                    N1=NODE1(CLINK)
                    N2=NODE2(CLINK)
                    IF ( TROAD(CLINK).EQ.1 )  THEN
                      IF ( EXIT(N1).EQ.0 .AND. EXIT(N2).EQ.0 )  THEN
                        IF ( LINK(1).NE.CLINK )  THEN
                          LINK(N)=CLINK
                        ENDIF
                      ELSE
                        LINK(3)=CLINK
                      ENDIF
                    ENDIF
                  ENDIF
120                     CONTINUE
110                   CONTINUE
            ELSE
              IF ( QUAN.EQ.1 )  THEN
                DO 130 J=1,4
                  CLINK=ATTLIN(CNODE,J)
                  IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                    IF ( TROAD(CLINK).EQ.1 )  THEN
                      LINK(1)=CLINK
                    ENDIF
                  ENDIF
130                     CONTINUE
              ELSE
                DO 140 J=1,4
                  CLINK=ATTLIN(CNODE,J)
                  IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                    IF ( TROAD(CLINK).EQ.1 )  THEN
                      LINK(1)=CLINK
                    ENDIF
                  ENDIF
140                     CONTINUE
*
                DO 150 J=1,4
                  CLINK=ATTLIN(CNODE,J)
                  IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
```

```
                N1=NODE1(CLINK)
                N2=NODE2(CLINK)
                IF ( NTYPE(N1).EQ.2 .OR. NTYPE(N2).EQ.2 )  THEN
                  CALL CULCHK(CLINK,CUL,M1,M2,M3,A)
                  IF ( CUL.EQ.0 )  THEN
                    LINK(2)=CLINK
                  ELSE
                    LINK(3)=CLINK
                  ENDIF
                ELSE
                  LINK(3)=CLINK
                ENDIF
              ENDIF
150                 CONTINUE
          ENDIF
        ENDIF
   ELSE
        DO 160 N=1,QUAN
          DO 170 J=1,4
          CLINK=ATTLIN(CNODE,J)
          IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
            N1=NODE1(CLINK)
            N2=NODE2(CLINK)
            IF ( NTYPE(N1).EQ.2 .OR. NTYPE(N2).EQ.2 )  THEN
              CALL CULCHK(CLINK,CUL,M1,M2,M3,A)
              IF ( LINK(1).NE.CLINK .AND. CUL.EQ.0 )  THEN
                LINK(N)=CLINK
              ELSE
                LINK(3)=CLINK
              ENDIF
            ELSE
              LINK(3)=CLINK
            ENDIF
          ENDIF
170               CONTINUE
160             CONTINUE
        ENDIF
*
*
*
   IF ( LINK(1).EQ.0 .AND. QUAN.EQ.1 )  THEN
        LINK(1)=LINK(3)
   ENDIF
*
   IF ( LINK(2).EQ.0 .AND. LINK(1).NE.0 .AND. QUAN.EQ.2 )  THEN
        LINK(2)=LINK(3)
   ENDIF
*
*
   IF ( LINK(2).EQ.0 .AND. LINK(1).EQ.0 .AND. QUAN.EQ.2 )  THEN
        DO 180 N=1,QUAN
          DO 190 J=1,4
            CLINK=ATTLIN(CNODE,J)
            IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
              IF ( LINK(1).NE.CLINK )  THEN
                LINK(N)=CLINK
              ENDIF
```

```
                ENDIF
190                   CONTINUE
180                 CONTINUE
   ENDIF
*
*
*
   IF ( NTYPE(CNODE).EQ.5 )  THEN
          DO 200 J=1,4
            CLINK=ATTLIN(CNODE,J)
            IF ( CLINK.NE.0 )  THEN
              IF ( LINK(1).NE.CLINK )  THEN
                LINK(1)=CLINK
              ENDIF
            ENDIF
200                 CONTINUE
   ENDIF
*
   RETURN
   END
```

B.3.21 SUBROUTINE: DMAKER2

```
      SUBROUTINE DMAKER2(CNODE,QUAN,YEILD,STOP,LINK)


*****************************************************************************
*
*                                DECISION MAKER
*
*****************************************************************************


      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &       CNODE,QUAN,LINK(3),ATTLIN,TROAD
        &       ,YEILD,STOP,CLINK

      COMMON
        &       /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
        &       RCURB
        &       /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
        &       UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
        &       /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
        &       /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
        &       /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
*
*
*
*  set the local link to zero
      LINK(1)=0
      LINK(2)=0
      LINK(3)=0
*
*
*
*  determine the local link
      IF ( YEILD.EQ.1 )  THEN
              IF ( STOP.EQ.0 )  THEN
                DO 100 N=1,QUAN
                  DO 110 J=1,4
                    CLINK=ATTLIN(CNODE,J)
                    IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                      IF ( TROAD(CLINK).EQ.2 )  THEN
                        IF ( LINK(1).NE.CLINK )  THEN
                            LINK(N)=CLINK
                        ENDIF
                      ENDIF
                    ENDIF
110                       CONTINUE
100                     CONTINUE
              ELSE
                DO 115 J=1,4
                  CLINK=ATTLIN(CNODE,J)
```

```
                IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                  IF ( TROAD(CLINK).EQ.2 )  THEN
                    LINK(1)=CLINK
                  ELSE
                    LINK(2)=CLINK
                  ENDIF
                ENDIF
115                   CONTINUE
            ENDIF
   ELSE
            DO 120 N=1,QUAN
              DO 130 J=1,4
                CLINK=ATTLIN(CNODE,J)
                IF ( CLINK.NE.0 .AND. WROAD(CLINK).NE.0.0 )  THEN
                  IF ( TROAD(CLINK).EQ.1 )  THEN
                    IF ( LINK(1).NE.CLINK )  THEN
                      LINK(N)=CLINK
                    ENDIF
                  ENDIF
                ENDIF
130                   CONTINUE
120                 CONTINUE
   ENDIF
*
   RETURN
   END
```

B.3.22 SUBROUTINE: DRAIN

```
      SUBROUTINE DRAIN(CNODE,CLINK,FLAG)


*****************************************************************************
*
*                                    DRAIN
*
*****************************************************************************
*
*
*
*  CNODE  - current node in the memory.
*  PNODE  - previous node in the memory.
*  ATTLIN - the links associate with the given node. ( NOTE: max. links can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  FLAG   - a flag indicates the direction of the drainage flow.
*           when  FLAG = 0  means flow out
*                 FLAG = 1  means flow in
*  NODE1  - origin node of the current link
*  NODE2  - dest. node of the current link
*  ELEV   - ground elevation of a given node
*  LINK   - the link associated with type 2 node
*  WROAD  - the road width of a current link
*  RADIUS - the radius of the cul-de-sac/bulb curve/tube sac
*
*
*
      PARAMETER
     &      NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &      CNODE,PNODE,CLINK,FLAG,ATTLIN

      REAL
     &      LLINK,LROAD

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
*
*
*
*  set the flow flag to negative one
      FLAG=-1
*  determine the previous node
      IF ( NODE1(CLINK).EQ.CNODE )  THEN
             PNODE=NODE2(CLINK)
      ELSE
```

```
            PNODE=NODE1(CLINK)
      ENDIF
*     if the current node is type 1, and the previous node is type 2
*     then this is a cul-de-sac, since this type 2 node will
*     not have any elevation.  Therefore, we must trace to another
*     previous node
      IF ( NTYPE(CNODE).EQ.1 )  THEN
            IF ( RADIUS(CNODE).NE.0.0 .AND. NTYPE(PNODE).EQ.2 )  THEN
              N=PNODE
              DO 100 J=1,4
                LINK=ATTLIN(N,J)
                IF ( LINK.EQ.0 .OR. WROAD(LINK).EQ.0.0 )  GOTO 100
                IF ( CLINK.NE.LINK )  THEN
                  IF ( NODE1(LINK).EQ.N )  THEN
                    PNODE=NODE2(LINK)
                  ELSE
                    PNODE=NODE1(LINK)
                  ENDIF
                ENDIF
100                   CONTINUE .
            ENDIF
      ENDIF
*     compare the elevation of the origin node and dest. node
*     determine the drainage flow condition
      IF ( ELEV(CNODE).GT.ELEV(PNODE) )  FLAG=0
      IF ( ELEV(CNODE).LT.ELEV(PNODE) )  FLAG=1
*
*
*
      RETURN
      END
```

B.3.23 SUBROUTINE: DSSDPTH

```
      SUBROUTINE DSSDPTH(CLINK,DPTH1,DPTH2)


*****************************************************************
*                                                               *
*                        SUBROUTINE DSSDPTH                     *
*                                                               *
*****************************************************************
*
*
*
*  XXXXXX -
*  ATTLIN - the links associate with the given node. ( NOTE: max. link can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory.
*  DS     - subscript.
*  DSLINK - domestic sewer elev. of a given link.
*  NODE1  - origin node of the current link.
*  NODE2  - dest. node of the current link.
*  DPTH1  - the depth of excavation at origin node.
*  DPTH2  - the depth of excavation at dest. node.
*  J      - loop counter
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  ATTLIN,CNODE,CLINK,DS

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM)
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &      /AREA32/SSLINK(NODNUM,4),AVEDEP(LINNUM),DSLINK(NODNUM,4)
*
*
*
*  save the origin node
      CNODE=NODE1(CLINK)
*
*  find the correct subscript to store the current link's
*  domestic sewer elev.
      DO 100 J=1,4
              IF ( ATTLIN(CNODE,J).EQ.CLINK )  THEN
                      DS=J
              ENDIF
100           CONTINUE
      DSLINK(CNODE,DS)=ELEV(CNODE)-DPTH1
```

```
*
*
*
*  save dest node
   CNODE=NODE2(CLINK)
*
*  find the correct subscript to store the current link's
*  domestic sewer elev.
   DO 110 J=1,4
            IF ( ATTLIN(CNODE,J).EQ.CLINK )  THEN
                     DS=J
            ENDIF
110         CONTINUE
   DSLINK(CNODE,DS)=ELEV(CNODE)-DPTH2
*
*
*
   RETURN
   END
```

B.3.24 SUBROUTINE: FINDPATH

```
      SUBROUTINE FINDPATH(N,J,C)


      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,ATTLIN,PATH,
        &      S,Z,C,EASE

      REAL
        &  LLINK,LROAD,LENGTH,KM

      COMMON
        &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
        &      ANGLE(NODNUM)
        &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
        &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
        &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
        &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &      /AREA26/ADJWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
        &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
        &      /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)

*     CLEAR THE EASEMENT FLAG
              EASE=0

*     STARTING NODE
10            S=N

*     COUNTER ( NUMBER OF THE LINKS)
              C=0

*     CURRENT LINK
              L=ATTLIN(S,J)

              IF ( L.EQ.0 )  GOTO 110
*     NO SUCH LINK EXIST

50            IF ( WROAD(L).EQ.0.0 .AND. C.EQ.0 )  THEN
*     DETECTED AN EASEMENT ( ATTACHED TO THE TYPE 3,4)
                      J=J+1
                      IF ( J.GE.5 )  THEN
                              EASE=0
                              GOTO 110
                      ELSE
                              EASE=1
                              GOTO 10
                      ENDIF
              ENDIF

              C=C+1
              PATH(N,J,C)=L

              IF ( NODE1(L).EQ.S )  THEN
```

```
                S=NODE2(L)
        ELSE
                S=NODE1(L)
        ENDIF

        IF ( NTYPE(S).NE.2 )  GOTO 110
*  TYPE 2 NODE COMES HERE

*  DETERMINE THE NEXT LINK ( DOES EASEMENT ATTACH TO THIS NODE? )
   DO 140 NN=1,4

        IF ( ATTLIN(S,NN).EQ.0 )  GOTO 140

        IF ( ATTLIN(S,NN).NE.L )  THEN
                IF ( WROAD(ATTLIN(S,NN)).NE.0.0 )  THEN
                        Z=NN
                ENDIF
        ENDIF

140     CONTINUE

        L=ATTLIN(S,Z)

        GOTO 50

110     IF ( EASE.EQ.1 )  THEN

*  EXCHANGE

        DO 120 MM=1,C
                PATH(N,J-1,MM)=PATH(N,J,MM)
120             CONTINUE
        J=J-1

   ENDIF

   RETURN
   END
```

B.3.25 **SUBROUTINE: GOGRAD**

```
      SUBROUTINE GOGRAD (BASE,GROWTH,INTRST,N,PWORTH,AWORTH,FWORTH)

      REAL INTRST

      RATE=INTRST/100.0
      GDRATE=GROWTH/100.0

*
      IF ( GDRATE.EQ.RATE )  THEN
              PWORTH=BASE*N/(1.0+GDRATE)
      ELSE
              IF ( GDRATE.GT.RATE )  THEN
                W=(1.0+GDRATE)/(1.0+RATE)-1.0
                PWORTH=BASE/(1.0+RATE)*(((1.0+W)**N-1.0)/W)
              ELSE
                W=(1.0+RATE)/(1.0+GDRATE)-1.0
                PWORTH=BASE/(1.0+GDRATE)*(((1.0+W)**N-1.0)/(W*((1.0+W)**N)))
              ENDIF
      ENDIF
*
*     find the future worth
      FWORTH=PWORTH*(1.0+RATE)**N
*
*     find the annual worth
      AWORTH=UNFORM(INTRST,N,PWORTH)


      RETURN
      END
```

B.3.26 SUBROUTINE: GRAPH

```
      SUBROUTINE GRAPH(REVMAX,REVMIN,TCOS)

*****************************************************************
*                                                               *
*                              GRAPH                            *
*                                                               *
*****************************************************************
*
*
*
*  XSPA   - number of spaces in each horizontal section.
*  XSEC   - number of horizontal sections.
*  XTOTAL - horizontal spacing counter.
*  YSPA   - number of spaces in each vertical section.
*  YSEC   - number of vertical sections.
*  YTOTAL - vertical spacing counter.
*  MROW   - max. rows.
*  MCOL   - max. columns.
*  COL    - starting column.
*  LENGTH - length of the phase/sentence.
*  SLOPE  - slope of the graph.
*  FRONT  - number of spaces in front of the phase/sentence.
*  LINE   - line characters
*  YNAME  - vertical phase/sentence of the graph.
*  XNAME  - horizontal phase/sentence of the graph.
*  TITLE  - title of the graph.
*  SIDE   - one character.
*  TCOS   - (total construction cost+assume cost) / total frontage ( $/m )
*  REVMAX - max. revenue. ( $/m )
*  REVMIN - min. revenue. ( $/m )
*  INVMAX - max. investment rate of return.
*  INVMIN - min. investment rate of return.
*  XNUMB - horizontal numbering system.
*  XVAL   - revenue per horizontal section.
*  XVAL1  - revenue per column.
*  YVAL   - investment rate of return per vertical section.
*  YVAL1  - investment rate of return per row.
*  X,Y,I,
*  J,L,K  - loop counter
*
*
*
      INTEGER Y,X,COL,YSPA,XSPA,YTOTAL,XTOTAL,YSEC,XSEC,
     &        FRONT,WUNIT1

      REAL INVMAX,INVMIN,XNUMB(15)

      CHARACTER LINE*132,YNAME*55,XNAME*132,TITLE*132,SIDE*1

      COMMON
     &        /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2

      TITLE='INVESTMENT RATE OF RETURN  versus  REVENUE '
      XNAME='REVENUE ( $ per meter ) '
```

```
      YNAME='INVESTMENT RATE OF RETURN '% ' '
*
*     control variables
      COL=11
      YSPA=5
      XSPA=10
      XSEC=10
      YSEC=10
*
*
*     find the max/min Y-value
      INVMIN=(REVMIN-TCOS)/TCOS*100
      INVMAX=(REVMAX-TCOS)/TCOS*100
*
*     set up the coordinate system
      MROW=YSPA*YSEC
      MCOL=XSPA*XSEC
      XVAL=(REVMAX-REVMIN)/(XSEC)
      XVAL1=(REVMAX-REVMIN)/MCOL
      YVAL=(INVMAX-0.0)/(YSEC)
      YVAL1=(INVMAX-0.0)/MROW
*
*
*
*     find the length of the title, and the starting point
      LENGTH=LEN('INVESTMENT RATE OF RETURN  versus  REVENUE ')
      FRONT=INT((MCOL-LENGTH)/2.0)+1
      DO 90 I=1,FRONT
              TITLE=' '//TITLE
   90 CONTINUE
      WRITE (WUNIT1,205)
      WRITE (WUNIT1,230) TITLE
      WRITE (WUNIT1,240)
*
*
*
*     50 x 120
*     find the length of the phase, and the starting point
      LENGTH=LEN('INVESTMENT RATE OF RETURN '% ' ')
      FRONT=MROW-INT((MROW-LENGTH)/2.0)+1
*
      YTOTAL=YSPA*YSEC
      J=0
      L=0
      DO 100 Y=MROW+1,1,-1
              DO 110 I=COL,132
                LINE(I:I)=' '
  110                 CONTINUE
*
*     draw the vertical grid
*
              XTOTAL=XSPA+COL
              DO 113 X=COL+1,MCOL+COL+1
                IF ( X.EQ.XTOTAL )  THEN
                  LINE(X:X)=':'
                  XTOTAL=XTOTAL+XSPA
                ENDIF
```

```
113                 CONTINUE
*
*  draw the Y-axis
*           print the vertical phase
            IF ( Y.LE.FRONT )  THEN
              L=L+1
              IF ( L.LT.LENGTH )  THEN
                SIDE(1:1)=YNAME(L:L)
              ELSE
                SIDE(1:1)=' '
              ENDIF
            ELSE
              SIDE(1:1)=' '
            ENDIF
*
            IF ( Y.EQ.YTOTAL ) THEN
*  draw the horizontal grid
              XTOTAL=XSPA+COL
              DO 115 K=COL+1,MCOL+COL+1
                IF ( K.EQ.XTOTAL )  THEN
                  LINE(K:K)='+'
                  XTOTAL=XTOTAL+XSPA
               ELSE
                  LINE(K:K)='-'
               ENDIF
115                   CONTINUE
              VAL=(INVMAX)-YVAL*J
              LINE(COL:COL)='+'
              YTOTAL=YTOTAL-YSPA
              J=J+1
            ELSE
              LINE(COL:COL)='!'
              VAL=0
            ENDIF
*
*  calculate the coordinate
*
            DO 120 I=0,MCOL
              RATE=(REVMIN+I*XVAL1-TCOS)/TCOS*100
              IF ( INT(RATE/YVAL1).EQ.(Y) )  THEN
                LINE(I+COL:I+COL)='*'
              ENDIF
120                 CONTINUE
*
*
            IF ( VAL.NE.0.0 )  THEN
              WRITE (WUNIT1,210) SIDE,VAL,LINE
            ELSE
              WRITE (WUNIT1,200) SIDE,LINE
            ENDIF
100         CONTINUE
*
*
*
*  draw the X-axis
   SIDE(1:1)=' '
   XTOTAL=XSPA+COL
```

```
      LINE(COL:COL)='L'
      DO 150 X=COL+1,MCOL+COL+1
              IF ( X.EQ.XTOTAL )  THEN
                LINE(X:X)=':'
                XTOTAL=XTOTAL+XSPA
              ELSE
                LINE(X:X)='.'
              ENDIF
150           CONTINUE
      WRITE(WUNIT1,210) SIDE,0.0,LINE
*
      DO 160 I=1,XSEC+1
              XNUMB(I)=REVMIN+(I-1)*XVAL
160           CONTINUE
      WRITE (WUNIT1,220) (XNUMB(I),I=1,XSEC+1)
*
*  find the length of the phase, and the starting point
      LENGTH=LEN('REVENUE ( $ per meter ) ')
      FRONT=INT((MCOL-LENGTH)/2.0)+1
      DO 170 I=1,FRONT
              XNAME=' '//XNAME
170           CONTINUE
      WRITE (WUNIT1,230) XNAME
*
*
*
      SLOPE=(INVMAX-INVMIN)/(REVMAX-REVMIN)/100
*
*
*  output the slope
      WRITE (WUNIT1,250) SLOPE
*
*
*
200           FORMAT (1X,1X,A1,8X,A122)
205           FORMAT ('1')
210           FORMAT (1X,1X,A1,1X,F6.1,1X,A122)
220           FORMAT (1X,7X,15(F6.0,4X))
230           FORMAT (1X,/,10X,A122)
240           FORMAT (/)
250           FORMAT (1X,//,15X,' IRR =',1X,G11.4,1X,'x (REVENUE) - 1')
*
*
*
      RETURN
      END
```

B.3.27 SUBROUTINE: GROUP1

```
      SUBROUTINE GROUP1(NUMLNK,LINKS,CHONUM,CHOLNK,NUMNOD,TBTRY)
*     ************************************************************
*     *    This subroutine associates 'from' and 'to' links in    *
*     *              groups and then sorts the groups.            *
*     ************************************************************
*     * NLINK =number of links.
*     * NTYPE =type of node.
*     * ELEV  =ground elevations of nodes.
*     * NODE1 =high node of link.
*     * NODE2 =low node of link.
*     * REFNOD=node currently being referenced.
*     * REFLNK=link currently being referenced.
*     * CHONUM=number of choices made in determining correct associations.
*     * PATHS =link numbers of links adjacent to REFNOD.
*     * X,X1-X7,V,V1,V2=miscellaneous use.
*     * CHECK =holds which links are in system being evaluated.
*     * SRTIDX=index into sort array.
*     * SORT  =array for sorting.
*     * LIST1,LIST2=arrays for sorting.
*     * NUMNOD=number of nodes involved in system being evaluated.
*     * NODES =which nodes are in system being evaluated.
*     *         -(index,1-5).
*     *         -1 is the identity of the node.
*     *         -(2-4) are the 'from' links.
*     *         -5 is the 'to' link.
*     * ERRFLG=flag indicating how error messages should be delivered.
*     * FLAG  =flag to indicate when a sort cycle is finished.
*     * NUMEXT=number of subdivision exits in the system being evaluated.
*     * EXTREF=names of nodes at which exits occur.
*     * NUMLNK=number of links in system.
*     * LINKS =links involved in system.
*     * TBTRY =values returned to calling routine.
*     * NEWLEN=the length of the optimum path accessed by the current
*     *         link.
*     * OLDLEN=the length of the optimum path accessed by the link which
*     *         previously accessed the overall optimum path (which may
*     *         replaced by NEWLEN if NEWLEN is shorter).
*     * CHEKIN=used to mark off links used in the creation of groups.

      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  REFNOD,CHONUM,CHOLNK(25,3),PATHS(6),CHECK(150),SRTIDX,
     &  SORT(150,5),X,X1,X2,X3,X4,X5,X6,X7,V,N,LIST1(50),LIST2(50),
     &  NUMNOD,NODES(150,5),FLAG,NUMEXT,EXTREF(25),NUMLNK,LINKS(150),
     &  TBTRY(150,4),RUNIT1,RUNIT2,WUNIT1,WUNIT2,REFLNK,CHEKIN(150),
     &  NUMPTH

      REAL V1,V2,NEWLEN,OLDLEN

      COMMON
     &     /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
     &     /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &     UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
```

```
      &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2

*     ***
*     * Clear CHECK.
*     ***
   DO 10, L=1, 150
   CHECK(L)=0
10 CONTINUE

*     ***
*     * Search for nodes which are accessed by more than one
*     * link in the system.
*     ***
   DO 20, L1=1, NNODE
   X1=0
   DO 30, L2=1, NUMLNK
   X2=NODE1(LINKS(L2))
   X3=NODE2(LINKS(L2))
   IF((L1.EQ.X2).OR.(L1.EQ.X3))X1=X1+1
30 CONTINUE
   IF(X1.GT.1)THEN
    CHECK(L1)=1
    NUMNOD=NUMNOD+1
    NODES(NUMNOD,1)=L1
   ENDIF
20 CONTINUE

*     ***
*     * Search for nodes in the system which are adjacent to an
*     * exit, and which haven't already been included in the system.
*     ***
   DO 40, L=1, NUMLNK
   X1=NODE1(LINKS(L))
   X2=NODE2(LINKS(L))
   IF((NTYPE(X1).LT.0).AND.(CHECK(X1).EQ.0))THEN
    CHECK(X1)=1
    NUMNOD=NUMNOD+1
    NODES(NUMNOD,1)=X1
   ENDIF
   IF((NTYPE(X2).LT.0).AND.(CHECK(X2).EQ.0))THEN
    CHECK(X2)=1
    NUMNOD=NUMNOD+1
    NODES(NUMNOD,1)=X2
   ENDIF
40 CONTINUE

*     ***********
*     * Establish the relationships of links with respect to the nodes
*     * involved in the system.
*     ***********

*     ******
*     * Count subdivision exits in the system.
*     ******
   DO 50, L=1, NUMNOD
   IF(NTYPE(NODES(L,1)).LT.0)THEN
    NUMEXT=NUMEXT+1
```

```
      EXTREF(NUMEXT)=L
     ENDIF
   50 CONTINUE

*     ******
*     * Create group data for exit nodes.
*     ******
     X1=0
     DO 60, L1=1, NUMEXT
      REFNOD=NODES(EXTREF(L1),1)
      CALL PATHS7(NODE1,NODE2,REFNOD,NUMLNK,LINKS,PATHS)
      DO 70, L2=1, 3
      IF((PATHS(L2).NE.0).AND.(CHEKIN(PATHS(L2)).EQ.0))THEN
       NODES(EXTREF(L1),L2+1)=PATHS(L2)
       CHEKIN(PATHS(L2))=1
       X1=X1+1
       LIST1(X1)=PATHS(L2)
      ENDIF
   70 CONTINUE
      NODES(EXTREF(L1),5)=150
   60 CONTINUE

*     ***
*     * Formulate the remainder of the groups until all links are used.
*     ***
     DO WHILE(FLAG.EQ.0)

     X2=0

*     * Use LIST1 to form new groups.  Use LIST1 to form LIST2.
*     * Copy LIST2 to LIST1 and start again.
     DO 80, L1=1, X1
     REFLNK=LIST1(L1)
     CALL PATHS2(NUMLNK,LINKS,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
       &  PATHS)
     IF(NUMPTH.NE.0)THEN
      X3=NODE1(REFLNK)
      X4=NODE2(REFLNK)
      X5=NODE1(PATHS(1))
      X6=NODE2(PATHS(1))
      IF((X3.EQ.X5).OR.(X3.EQ.X6))REFNOD=X3
      IF((X4.EQ.X5).OR.(X4.EQ.X6))REFNOD=X4
      DO 90, L2=1, NUMNOD
       IF(REFNOD.EQ.NODES(L2,1))X7=L2
   90 CONTINUE
      NODES(X7,5)=REFLNK
      DO 100, L2=2, 4
      IF(PATHS(L2-1).NE.0)THEN
       NODES(X7,L2)=PATHS(L2-1)
       CHEKIN(PATHS(L2-1))=1
       X2=X2+1
       LIST2(X2)=PATHS(L2-1)
      ENDIF
  100         CONTINUE
     ENDIF
   80 CONTINUE
```

```
*     * Clear LIST1 and copy LIST2 into LIST1.
   DO 110, L=1, 50
   LIST1(L)=0
   IF(L.LE.X2) LIST1(L)=LIST2(L)
110         CONTINUE

   IF(X2.EQ.0)FLAG=1

   X1=X2

   ENDDO

*     ***********
*     * Delete any NODES entry which has no 'to' link.
*     * If an entry needs to be deleted, move all other
*     * entries down one positon.
*     ***********
   FLAG=0
   DO WHILE(FLAG.EQ.0)
   FLAG=1
   DO 120, L2=1, NUMNOD-1
   IF(NODES(L2,5).EQ.0)THEN
    FLAG=0
    DO 130, L3=L2, NUMNOD-1
    NODES(L3,1)=NODES(L3+1,1)
    NODES(L3,2)=NODES(L3+1,2)
    NODES(L3,3)=NODES(L3+1,3)
    NODES(L3,4)=NODES(L3+1,4)
    NODES(L3,5)=NODES(L3+1,5)
130          CONTINUE
    NUMNOD=NUMNOD-1
   ENDIF
120         CONTINUE
   ENDDO

*     * Check the last entry (not accessed by above routine).
   IF(NODES(NUMNOD,5).EQ.0)NUMNOD=NUMNOD-1

*     ******
*     * Move filled 'from' positions towards NODE( ,2) and zeroes
*     * towards NODE( ,4).
*     ******
   DO 140, L=1, NUMNOD

   IF(NODES(L,2).EQ.0)THEN
    NODES(L,2)=NODES(L,3)
    NODES(L,3)=NODES(L,4)
    NODES(L,4)=0
   ENDIF

   IF(NODES(L,2).EQ.0)THEN
    NODES(L,2)=NODES(L,3)
    NODES(L,3)=0
   ENDIF

   IF(NODES(L,3).EQ.0)THEN
    NODES(L,3)=NODES(L,4)
```

```
      NODES(L,4)=0
     ENDIF

140          CONTINUE

*     ******
*     * Count subdivision exits in the system.
*     ******
     NUMEXT=0
     DO 150, L=1, NUMNOD
     IF(NTYPE(NODES(L,1)).LT.0)THEN
      NUMEXT=NUMEXT+1
      EXTREF(NUMEXT)=L
     ENDIF
150          CONTINUE

*     ***********
*     * Sort the groups of associated links into continuous subsystems
*     * of the total system (prerequisite for flow calculations).
*     ***********

*     ******
*     * Clear CHECK.
*     ******
     DO 160, L=1, 150
     CHECK(L)=0
160          CONTINUE

*     ******
*     * Begin at each exit and trace backwards until the order
*     * of a subsystem is defined, copying associated groups of
*     * links in the correct order into a sorting array until
*     * all the subsystems are defined, then copy the SORT array
*     * into TRTRY.
*     ******
     DO WHILE(NUMEXT.NE.0)

*     * Clear the flag, put the exit group identifier in LIST1,
*     * and mark off this exit group in CHECK.
     FLAG=0
     LIST1(1)=EXTREF(NUMEXT)
     CHECK(EXTREF(NUMEXT))=1
     X1=1

*     * Increment the index and store the exit group in SORT.
       SRTIDX=SRTIDX+1
       SORT(SRTIDX,1)=NODES(LIST1(1),1)
       SORT(SRTIDX,2)=NODES(LIST1(1),2)
       SORT(SRTIDX,3)=NODES(LIST1(1),3)
       SORT(SRTIDX,4)=NODES(LIST1(1),4)
       SORT(SRTIDX,5)=NODES(LIST1(1),5)


*     ***
*     * Trace until the flag is set.
*     ***
     DO WHILE(FLAG.EQ.0)
```

```
      X2=0

*       * Move through LIST1.
      DO 170, L1=1, X1

*       * Move through 'from' links.
      DO 180, L2=2, 4

      V=NODES(LIST1(L1),L2)

      IF(V.NE.0)THEN

*       * Search for an unused NODES entry which accesses the
*       * node defined by NODES(LIST1(L1),1).
       X3=0
       DO 190, L3=1, NUMNOD
       X4=NODES(L3,5)
       X5=NODES(LIST1(L1),L2)
       X6=CHECK(L3)
       IF((X4.EQ.X5).AND.(X6.EQ.0))X3=L3
190          CONTINUE
       IF(X3.NE.0)CHECK(X3)=1

*       * If X3<>0, copy the info about the group into SORT, increment
*       * the LIST2 index, and store the reference value in LIST2.
       IF(X3.NE.0)THEN
        SRTIDX=SRTIDX+1
        SORT(SRTIDX,1)=NODES(X3,1)
        SORT(SRTIDX,2)=NODES(X3,2)
        SORT(SRTIDX,3)=NODES(X3,3)
        SORT(SRTIDX,4)=NODES(X3,4)
        SORT(SRTIDX,5)=NODES(X3,5)
        X2=X2+1
        LIST2(X2)=X3
       ENDIF

      ENDIF

180         CONTINUE

170         CONTINUE

*       * Clear LIST1 and copy LIST2 into LIST1.
      DO 200, L=1, 50
      LIST1(L)=0
      IF(L.LE.X2) LIST1(L)=LIST2(L)
200         CONTINUE

*       * Check for the second list being empty:  if so, set FLAG=1.
      IF(X2.EQ.0)FLAG=1

      X1=X2

      ENDDO

*       * Update NUMEXT.
```

```
      NUMEXT=NUMEXT-1

      ENDDO

*       ******
*       * Copy the sorted array SORT into TBTRY.  Since SORT
*       * contains the groups in order from exit-group to inner-group,
*       * (backwards from what the flow/slope routine wants), the
*       * entries must be copied in reverse.
*       ******
      DO 210, L=SRTIDX, 1, -1

      X=SRTIDX-L+1
      TBTRY(X,1)=SORT(L,2)
      TBTRY(X,2)=SORT(L,3)
      TBTRY(X,3)=SORT(L,4)
      TBTRY(X,4)=SORT(L,5)

*       * Set the 'to' links of exit groups to 150 (the 'to'
*       * link will be 0 if it's of an exit group).
      IF(TBTRY(X,4).EQ.0)TBTRY(X,4)=150

210         CONTINUE

      RETURN
      END
```

B.3.28 SUBROUTINE: GROUP2

```
      SUBROUTINE GROUP2(NUMLNK,LINKS,CHONUM,CHOLNK,NUMNOD,TBTRY)
*     ************************************************************
*     *    This subroutine associates 'from' and 'to' links in    *
*     *              groups and then sorts the groups.            *
*     ************************************************************
*     * NLINK =number of links.
*     * NTYPE =type of node.
*     * ELEV  =ground elevations of nodes.
*     * NODE1 =high node of link.
*     * NODE2 =low node of link.
*     * REFNOD=node currently being referenced.
*     * REFLNK=link currently being referenced.
*     * CHONUM=number of choices made in determining correct associations.
*     * CHOLNK=link identifiers of links a choice was made over.
*     *       -(index,1-3).
*     *       -1 is the name of the link not chosen.
*     *       -2 is the name of the link chosen.
*     *       -3 is the name of the node at which the choice was made.
*     * PATHS =link numbers of links adjacent to REFNOD.
*     * X,X1-X6,V,V1,V2=miscellaneous use.
*     * CHECK =holds which links are in system being evaluated.
*     * SRTIDX=index into sort array.
*     * SORT  =array for sorting.
*     * LIST1,LIST2=arrays for sorting.
*     * NUMNOD=number of nodes involved in system being evaluated.
*     * NODES =which nodes are in system being evaluated.
*     *        -(index,1-5).
*     *        -1 is the identity of the node.
*     *        -(2-4) are the 'from' links.
*     *        -5 is the 'to' link.
*     * FLAG  =flag to indicate when a sort cycle is finished.
*     * NUMEXT=number of subdivision exits in the system being evaluated.
*     * EXTREF=names of nodes at which exits occur.
*     * NUMLNK=number of links in system.
*     * LINKS =links involved in system.
*     * TBTRY =values returned to calling routine.
*     * NEWLEN=the length of the optimum path accessed by the current
*     *        link.
*     * OLDLEN=the length of the optimum path accessed by the link which
*     *        previously accessed the overall optimum path (which may
*     *        replaced by NEWLEN if NEWLEN is shorter).

      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  REFNOD,CHONUM,CHOLNK(25,3),PATHS(6),CHECK(150),SRTIDX,
     &  SORT(150,5),X,X1,X2,X3,X4,X5,X6,V,N,LIST1(50),LIST2(50),
     &  NUMNOD,NODES(150,5),FLAG,NUMEXT,EXTREF(25),NUMLNK,LINKS(150),
     &  TBTRY(150,4),RUNIT1,RUNIT2,WUNIT1,WUNIT2,REFLNK

      REAL V1,V2,NEWLEN,OLDLEN

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM)
```

```
      &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
      &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
      &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2

*     ***
*     * Clear CHECK.
*     ***
   DO 10, L=1, 150
   CHECK(L)=0
10 CONTINUE

*     ***
*     * Search for nodes which are accessed by more than one
*     * link in the system.
*     ***
   DO 20, L1=1, NNODE
   X1=0
   DO 30, L2=1, NUMLNK
   X2=NODE1(LINKS(L2))
   X3=NODE2(LINKS(L2))
   IF((L1.EQ.X2).OR.(L1.EQ.X3))X1=X1+1
30 CONTINUE
   IF(X1.GT.1)THEN
    CHECK(L1)=1
    NUMNOD=NUMNOD+1
    NODES(NUMNOD,1)=L1
   ENDIF
20 CONTINUE

*     ***
*     * Search for nodes in the system which are adjacent to an
*     * exit, and which haven't already been included in the system.
*     ***
   DO 40, L=1, NUMLNK
   X1=NODE1(LINKS(L))
   X2=NODE2(LINKS(L))
   IF((NTYPE(X1).LT.0).AND.(CHECK(X1).EQ.0))THEN
    CHECK(X1)=1
    NUMNOD=NUMNOD+1
    NODES(NUMNOD,1)=X1
   ENDIF
   IF((NTYPE(X2).LT.0).AND.(CHECK(X2).EQ.0))THEN
    CHECK(X2)=1
    NUMNOD=NUMNOD+1
    NODES(NUMNOD,1)=X2
   ENDIF
40 CONTINUE

*     ***
*     * Clear CHECK.
*     ***
   DO 50, L=1, 150
   CHECK(L)=0
50 CONTINUE

*     ***********
*     * Establish the relationships of links with respect to the nodes
```

```
*     * involved in the system.
*     **********

      DO 60, L1=1, NUMNOD

*     ******
*     * For each node in the system, call PATHS7 to find out what
*     * system links access the node.
*     ******
      REFNOD=NODES(L1,1)
      CALL PATHS7(NODE1,NODE2,REFNOD,NUMLNK,LINKS,PATHS)

*     * Copy PATHS into NODES.
      DO 70, L2=1,4
      NODES(L1,L2+1)=PATHS(L2)
70    CONTINUE

*     ******
*     * Organize 'from' and 'to' links.
*     ******
      DO 80, L2=2, 4

*     ***
*     * Move this 'from' link into the 'to' position if the 'to'
*     * position is empty and the low end of the link is lower
*     * than REFNOD.
*     ***
      IF(NODES(L1,5).EQ.0)THEN
       V1=ELEV(NODE2(NODES(L1,L2)))
       V2=ELEV(REFNOD)
       IF(V1.LT.V2)THEN
        NODES(L1,5)=NODES(L1,L2)
        NODES(L1,L2)=0
       ENDIF
      ENDIF

*     ***
*     * Switch positions of the current 'from' link and the 'to' link
*     * if this link will provide the more optimum path and if the
*     * low end of the link is lower than REFNOD.
*     ***
      IF((NODES(L1,5).NE.0).AND.(NODES(L1,L2).NE.0))THEN

       V1=ELEV(NODE2(NODES(L1,L2)))
       V2=ELEV(REFNOD)

*     * If the link gets cleared and has an end lower than REFNOD and
*     * is a more optimum a choice than the existing 'to' link, then
*     * switch positions.  OPTPTH finds the optimum choice of path.
       OLDLEN=0.0
       NEWLEN=0.0
       IF(V1.LT.V2)THEN
        REFLNK=NODES(L1,5)
        CALL OPTPTH(NUMLNK,LINKS,REFLNK,OLDLEN)
        REFLNK=NODES(L1,L2)
        CALL OPTPTH(NUMLNK,LINKS,REFLNK,NEWLEN)
        IF(NEWLEN.LT.OLDLEN)THEN
```

```
        X=NODES(L1,L2)
        NODES(L1,L2)=NODES(L1,5)
        NODES(L1,5)=X
      ENDIF
     ENDIF
    ENDIF

 80 CONTINUE

*     ******
*     * Check to see if there is more than one downsloping link
*     * (relative to REFNOD), and remove the extra downsloping links,
*     * increment CHONUM, and store the choice info in CHOLNK.
*     ******
    DO 90, L2=2, 4
    IF(NODES(L1,L2).NE.0)THEN
     X=NODE2(NODES(L1,L2))
     IF(ELEV(X).LT.ELEV(REFNOD))THEN
      CHONUM=CHONUM+1
      CHOLNK(CHONUM,1)=NODES(L1,L2)
      CHOLNK(CHONUM,2)=NODES(L1,5)
      CHOLNK(CHONUM,3)=REFNOD
      NODES(L1,L2)=0
     ENDIF
    ENDIF

 90 CONTINUE

*     ******
*     * Move filled 'from' positions towards NODE( ,2) and zeroes
*     * towards NODE( ,4).
*     ******
    IF(NODES(L1,2).EQ.0)THEN
     NODES(L1,2)=NODES(L1,3)
     NODES(L1,3)=NODES(L1,4)
     NODES(L1,4)=0
    ENDIF

    IF(NODES(L1,2).EQ.0)THEN
     NODES(L1,2)=NODES(L1,3)
     NODES(L1,3)=0
    ENDIF

    IF(NODES(L1,3).EQ.0)THEN
     NODES(L1,3)=NODES(L1,4)
     NODES(L1,4)=0
    ENDIF

 60 CONTINUE

*     ***********
*     * Delete any NODES entries which have no 'from' links and only a
*     * 'to' link.  If an entry needs to be deleted, move all other
*     * entries down one positon.
*     ***********
    FLAG=0
    DO WHILE(FLAG.EQ.0)
```

```
      FLAG=1
      DO 100, L1=1, NUMNOD-1
      X2=NODES(L1,2)
      X3=NODES(L1,3)
      X4=NODES(L1,4)
      IF((X2.EQ.0).AND.(X3.EQ.0).AND.(X4.EQ.0))THEN
       FLAG=0
       DO 110, L2=L1, NUMNOD-1
       NODES(L2,1)=NODES(L2+1,1)
       NODES(L2,2)=NODES(L2+1,2)
       NODES(L2,3)=NODES(L2+1,3)
       NODES(L2,4)=NODES(L2+1,4)
       NODES(L2,5)=NODES(L2+1,5)
110          CONTINUE
       NUMNOD=NUMNOD-1
      ENDIF
100          CONTINUE
      ENDDO

*       * Check the last entry (not accessed by above routine).
      X2=NODES(NUMNOD,2)
      X3=NODES(NUMNOD,3)
      X4=NODES(NUMNOD,4)
      IF((X2.EQ.0).AND.(X3.EQ.0).AND.(X4.EQ.0))NUMNOD=NUMNOD-1

*       ***********
*       * Sort the groups of associated links into continuous subsystems
*       * of the total system (prerequisite for flow calculations).
*       ***********

*       ******
*       * Clear CHECK.
*       ******
      DO 120, L=1, 150
      CHECK(L)=0
120          CONTINUE

*       ******
*       * Count subdivision exits in the system.
*       * If the central node of the system is an exit point,
*       * set the 'to' link equal to 0.
*       ******
      DO 130, L=1, NUMNOD
      IF(NTYPE(NODES(L,1)).LT.0)THEN
       NODES(L,5)=0
       NUMEXT=NUMEXT+1
       EXTREF(NUMEXT)=L
      ENDIF
130          CONTINUE

*       ******
*       * Begin at each exit and trace backwards until the order
*       * of a subsystem is defined, copying associated groups of
*       * links in the correct order into a sorting array until
*       * all the subsystems are defined, then copy the SORT array
*       * into TRTRY.
*       ******
```

```
      DO WHILE(NUMEXT.NE.0)

*       * Clear LIST1.
      DO 140, L=1, 25
      LIST1(L)=0
140         CONTINUE

*       * Clear the flag, put the exit group identifier in LIST1,
*       * and mark off this exit group in CHECK.
      FLAG=0
      LIST1(1)=EXTREF(NUMEXT)
      CHECK(EXTREF(NUMEXT))=1
      X1=1

*       * Increment the index and store the exit group in SORT.
        SRTIDX=SRTIDX+1
        SORT(SRTIDX,1)=NODES(LIST1(1),1)
        SORT(SRTIDX,2)=NODES(LIST1(1),2)
        SORT(SRTIDX,3)=NODES(LIST1(1),3)
        SORT(SRTIDX,4)=NODES(LIST1(1),4)
        SORT(SRTIDX,5)=NODES(LIST1(1),5)

*       ***
*       * Trace until the flag is set.
*       ***
      DO WHILE(FLAG.EQ.0)

      X2=0

*       * Move through LIST1.
      DO 150, L1=1, X1

*       * Move through 'from' links.
      DO 160, L2=2, 4

      V=NODES(LIST1(L1),L2)

      IF(V.NE.0)THEN

*       * Search for an unused NODES entry which accesses the
*       * node defined by NODES(LIST1(L1),1).
       X3=0
       DO 170, L3=1, NUMNOD
       X4=NODES(L3,5)
       X5=NODES(LIST1(L1),L2)
       X6=CHECK(L3)
       IF((X4.EQ.X5).AND.(X6.EQ.0))X3=L3
170         CONTINUE
       IF(X3.NE.0)CHECK(X3)=1

*       * If X3<>0, copy the info about the group into SORT, increment
*       * the LIST2 index, and store the reference value in LIST2.
       IF(X3.NE.0)THEN
        SRTIDX=SRTIDX+1
        SORT(SRTIDX,1)=NODES(X3,1)
        SORT(SRTIDX,2)=NODES(X3,2)
        SORT(SRTIDX,3)=NODES(X3,3)
```

```
      SORT(SRTIDX,4)=NODES(X3,4)
      SORT(SRTIDX,5)=NODES(X3,5)
      X2=X2+1
      LIST2(X2)=X3
     ENDIF

    ENDIF

160         CONTINUE

150         CONTINUE

*     * Clear LIST1 and copy LIST2 into LIST1.
    DO 180, L=1, 50
    LIST1(L)=0
    IF(L.LE.X2) LIST1(L)=LIST2(L)
180         CONTINUE

*     * Check for the second list being empty:  if so, set FLAG=1.
    IF(X2.EQ.0)FLAG=1

    X1=X2

    ENDDO

*     * Update NUMEXT.
    NUMEXT=NUMEXT-1

    ENDDO

*     ******
*     * Copy the sorted array SORT into TBTRY.  Since SORT
*     * contains the groups in order from exit-group to inner-group,
*     * (backwards from what the flow/slope routine wants), the
*     * entries must be copied in reverse.
*     ******
    DO 190, L=SRTIDX, 1, -1

    X=SRTIDX-L+1
    TBTRY(X,1)=SORT(L,2)
    TBTRY(X,2)=SORT(L,3)
    TBTRY(X,3)=SORT(L,4)
    TBTRY(X,4)=SORT(L,5)

*     * Set the 'to' links of exit groups to 150 (the 'to'
*     * link will be 0 if it's of an exit group).
    IF(TBTRY(X,4).EQ.0)TBTRY(X,4)=150

190         CONTINUE

    RETURN
    END
```

B.3.29 SUBROUTINE: OPTPTH

```
      SUBROUTINE OPTPTH(NSL,LINKS,REFLNK,OPTLEN)
*     ***********************************************
*     *   This subroutine identifies which link should  *
*     *   should be chosen when there is a choice, by   *
*     *   finding which one accesses the shortest path  *
*     *               downhill to an exit.              *
*     ***********************************************
*     * CHEKIN=keeps track of which links have been passed during
*     *        the test of the current link.
*     * PCHIDX=index into POSCHN.
*     * POSCHN=possible chain currently being checked.
*     * CHNIDX=index into CHAINS.
*     * CHAINS=list of chains which have been found.  Any chains
*     *        which lead to an exit are marked with a one in the
*     *        zero element.
*     * REFLNK=a pointer showing at which link the trace is at.
*     * BRANUM=the number of branches (forks in the path) passed.
*     * BRALNK=holds the link numbers before each branch.
*     * NUMPTH=number of possible paths test can follow.
*     * PATHS =a list of the link numbers of each of the paths adjacent
*     *        to REFLNK.
*     * FLAG1 =signifies an end to the test for a given link.
*     * FLAG2 =signifies a return to a former branch location
*     *        and recalculation of possible paths.
*     * RESULT=indicates whether the path choice to be made has
*     *        been made before.
*     * CHECK =used to mark off links which are obviously in the system.
*     * LENGTH=used to calculate the lenth of a chain.
*     * X,X1,X2=miscellaneous variables.
*     * V     =used for locating the chain of shortest length.
*     * OPTLEN=length of chosen links shortest (optimum) path to an exit.
*     * NSL   =number of system links.
*     * LINKS =links involved in the system.

      INTEGER
     &  TBTRYW,TBTRYD,TBTRYS,LINKS(150),
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,TROAD

      INTEGER CHEKED(150),CHEKIN(150),REFLNK,BRANUM,BRALNK(25),
     &  NUMPTH,PATHS(6),FLAG1,FLAG2,X1,X2,ERRFLG,CHECK(150),
     &  PCHIDX,POSCHN(10),CHNIDX,CHAINS(500,0:10),LNKTYP,RESULT,NSL

      REAL
     &  LLINK,LROAD,ELEV,MULTI,UNITS,V,LENGTH,CHAINL(500),OPTLEN

      COMMON
     &      /AREA2/NNODE,NTYPE(100),ELEV(100)
     &      /AREA3/NLINK,NODE1(150),NODE2(150),LLINK(150),
     &      UNITS(150),MULTI(150),AREA(150)
     &      /AREA4/LROAD(150),WROAD(150),TROAD(150)
     &  /AREA9/P(150),PDS(150),ASS(150)

      OPTLEN=0.0
      PCHIDX=1
      POSCHN(PCHIDX)=REFLNK
```

```
      CHNIDX=0
      FLAG1=0
      BRANUM=0
      CHEKIN(REFLNK)=1

*       * Find what possible paths there are for the test (initial call).
      CALL PATHS4(NLINK,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
     &  PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)

*       * Search until all chains have been determined.
      DO WHILE(FLAG1.EQ.0)

      FLAG2=0

*       ***
*       * Check the endpoints of the link for an exit.
*       * If the condition is met, increment CHNIDX, copy POSCHN
*       * to CHAINS, pull a branch ID from BRALNK, erase POSCHN backwards
*       * till it hits the ID (so from the branch on, the chain is new),
*       * set FLAG2 to idicate a skip of further calculations so that
*       * a new chain can be developed from the branch on.  If there aren't
*       * any branches left, set FLAG1.
*       ***
      X1=NTYPE(NODE1(REFLNK))
      X2=NTYPE(NODE2(REFLNK))

*       * Check for an exit.
      IF((X1.LT.0).OR.(X2.LT.0))THEN

*       * Increment CHNIDX and copy POSCHN into CHAINS.  Set the
*       * zero element to 1, since this chain reaches an exit.
       CHNIDX=CHNIDX+1
       DO 10, L2=1, PCHIDX
       CHAINS(CHNIDX,L2)=POSCHN(L2)
10     CONTINUE
       CHAINS(CHNIDX,0)=1

*       * Set FLAG1 if there aren't any branches left.
       IF(BRANUM.EQ.0) FLAG1=1

*       * If there are branches left, get the last branch and erase
*       * POSCHN back until that branch location (so a new chain can
*       * be built from there on).  Also, clear CHEKIN elements
*       * of erased POSCHN links.
       IF(BRANUM.NE.0)THEN
        REFLNK=BRALNK(BRANUM)
        BRANUM=BRANUM-1
        DO WHILE(POSCHN(PCHIDX).NE.REFLNK)
        CHEKIN(POSCHN(PCHIDX))=0
        PCHIDX=PCHIDX-1
        ENDDO
       ENDIF

*       * Set FLAG2 so further calculations are skipped;  start
*       * generating a new chain from the branch.
       FLAG2=1
```

```
      ENDIF

*       ***
*       * If there is more than one path, increment BRANUM, and
*       * store the identity of the link which lies before the branch.
*       ***
      IF(NUMPTH.GT.1)THEN
       BRANUM=BRANUM+1
       BRALNK(BRANUM)=REFLNK
      ENDIF

*       *  NOTE: from here on, if FLAG2=1, skip calculations.

*       ***
*       * If there are no paths, and another branch exists, take it.
*       * Set FLAG2 so further calculations are skipped.  The paths for the
*       * branch link are redetermined, and testing can restart from there.
*       * Increment CHNIDX and copy POSCHN into CHAINS, and erase POSCHN
*       * back to the branch (erasing CHEKIN entries along the way).
*       ***
      IF((FLAG2.EQ.0).AND.(NUMPTH.EQ.0))THEN

*       * Increment CHNIDX and copy POSCHN into CHAINS.
       CHNIDX=CHNIDX+1
       DO 20, L2=1, PCHIDX
       CHAINS(CHNIDX,L2)=POSCHN(L2)
20     CONTINUE

*       * Set FLAG1 if there aren't any branches left.
       IF(BRANUM.EQ.0) FLAG1=1

*       * If there are branches left, get the last branch and erase
*       * POSCHN back until that branch location (so a new chain can
*       * be built from there on).  Also, clear CHEKIN elements of
*       * erased POSCHN links.
       IF(BRANUM.NE.0)THEN
        REFLNK=BRALNK(BRANUM)
        BRANUM=BRANUM-1
        DO WHILE(POSCHN(PCHIDX).NE.REFLNK)
        CHEKIN(POSCHN(PCHIDX))=0
        PCHIDX=PCHIDX-1
        ENDDO
       ENDIF

*       * Set FLAG2 so further calculations are skipped;  start
*       * generating a new chain from the branch.
       FLAG2=1

      ENDIF

*       ***
*       * Take the first path on the list, if one exists (update REFLNK).
*       * Check the new link off as one which has been passed (update CHEKIN).
*       * Also update POSCHN.
*       ***
      IF((FLAG2.EQ.0).AND.(NUMPTH.NE.0))THEN
       REFLNK=PATHS(1)
```

```
      CHEKIN(REFLNK)=1
      PCHIDX=PCHIDX+1
      POSCHN(PCHIDX)=REFLNK
     ENDIF

*     ***
*     * Find what possible paths there are for the next test.
*     ***
     CALL PATHS4(NLINK,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
       & PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)

     ENDDO

*     ******
*     * Pick the shortest path and copy all its elements into
*     * CHEKED.
*     ******

*     ***
*     * Calculate the lengths for each chain which ends in an exit.
*     * Store the results in CHAINL.
*     ***
     DO 30, L2=1, CHNIDX
     LENGTH=0.0
     IF(CHAINS(L2,0).EQ.1)THEN
      X1=1
      DO WHILE((CHAINS(L2,X1).NE.0).AND.(X1.LE.10))
      LENGTH=LENGTH+LLINK(CHAINS(L2,X1))
      X1=X1+1
      ENDDO
     ENDIF
     CHAINL(L2)=LENGTH
30   CONTINUE

*     ***
*     * Search for a nonzero chain length.
*     ***
     X2=0
     DO 40, L2=1, 500
     IF(CHAINL(L2).NE.0.0)X2=L2
40   CONTINUE

*     ***
*     * Search for the shortest chain.
*     ***
     DO 50, L2=1, 500
     IF((CHAINL(L2).LT.CHAINL(X2)).AND.(CHAINL(L2).NE.0.0))X2=L2
50   CONTINUE

*     * Put the length of the optimum path into OPTLEN.
     IF((CHAINL(X2).LT.OPTLEN).OR.(OPTLEN.EQ.0.0))OPTLEN=CHAINL(X2)

*     ***
*     * Clear CHAINS.
*     ***
     DO 60, L2=1, 500
     DO 70, L3=0, 10
```

```
      CHAINS(L2,L3)=0
   70 CONTINUE
   60 CONTINUE

*        ***
*        * Clear CHAINL.
*        ***
      DO 80, L2=1, 500
      CHAINL(L2)=0.0
   80 CONTINUE

*        ***
*        * Clear CHEKIN.
*        ***
      DO 90, L2=1, 150
      CHEKIN(L2)=0
   90 CONTINUE

      RETURN

      END
```

B.3.30 **SUBROUTINE: PATHS1**

```
      SUBROUTINE PATHS1(NLINK,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
     &  PATHS)
*     ***************************************************
*     *  This subroutine finds what paths are adjacent  *
*     *    to a given link and which have not already   *
*     *                  been accessed.                 *
*     ***************************************************
*     * NLINK =number of links.
*     * NODE1,NODE2=node descriptors of links.
*     * REFLNK=link currently being considered.
*     * CHEKIN=array holding all links passed while 'checkin'.
*     * NUMPTH=number of paths possible paths found.
*     * PATHS =link numbers of possible paths.
*     * X1-X4 =miscellaneous use.

      INTEGER NLINK,NODE1(150),NODE2(150),REFLNK,CHEKIN(150),NUMPTH,
     &  PATHS(6),X1,X2,X3,X4

*     * Initialize variables.
      NUMPTH=0
      PATHS(1)=0
      PATHS(2)=0
      PATHS(3)=0
      PATHS(4)=0
      PATHS(5)=0
      PATHS(6)=0

      DO 10, L=1, NLINK
      X1=NODE1(REFLNK)
      X2=NODE2(REFLNK)
      X3=NODE1(L)
      X4=NODE2(L)
      IF((X1.EQ.X3).OR.(X1.EQ.X4).OR.(X2.EQ.X3).OR.(X2.EQ.X4))THEN
       IF(CHEKIN(L).EQ.0)THEN
        NUMPTH=NUMPTH+1
        PATHS(NUMPTH)=L
       ENDIF
      ENDIF

10    CONTINUE
      RETURN
      END
```

B.3.31 SUBROUTINE: PATHS2

```
      SUBROUTINE PATHS2(NUMLNK,LINKS,NODE1,NODE2,REFLNK,CHEKIN,NUMPTH,
     &  PATHS)
*     ***********************************************
*     *  This subroutine finds what paths are adjacent  *
*     *     to a given link, have not already been      *
*     *    accessed, and which are in the particular    *
*     *              system being checked.              *
*     ***********************************************
*     * NUMLNK=number of links.
*     * LINKS =links involved in system.
*     * NODE1,NODE2=node descriptors of links.
*     * REFLNK=link currently being considered.
*     * CHEKIN=array holding all links passed while 'checkin'.
*     * NUMPTH=number of paths possible paths found.
*     * PATHS =link numbers of possible paths.
*     * X1-X4 =miscellaneous use.

      INTEGER NUMLNK,LINKS(150),NODE1(150),NODE2(150),REFLNK,CHEKIN(150),
     &  NUMPTH,PATHS(6),X1,X2,X3,X4

*     * Initialize variables.
      NUMPTH=0
      PATHS(1)=0
      PATHS(2)=0
      PATHS(3)=0
      PATHS(4)=0
      PATHS(5)=0
      PATHS(6)=0

      DO 10, L=1, NUMLNK
      X1=NODE1(REFLNK)
      X2=NODE2(REFLNK)
      X3=NODE1(LINKS(L))
      X4=NODE2(LINKS(L))
      IF((X1.EQ.X3).OR.(X1.EQ.X4).OR.(X2.EQ.X3).OR.(X2.EQ.X4))THEN
       IF(CHEKIN(LINKS(L)).EQ.0)THEN
        NUMPTH=NUMPTH+1
        PATHS(NUMPTH)=LINKS(L)
       ENDIF
      ENDIF
   10 CONTINUE
      RETURN
      END
```

B.3.32 SUBROUTINE: PATHS3

```
      SUBROUTINE PATHS3(NLINK,NODE1,NODE2,WROAD,REFLNK,CHEKIN,
     &  NUMPTH,PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)
*     ***********************************************
*     *  This subroutine finds what paths are adjacent  *
*     * to a given link and which would, if chosen, not *
*     *  be duplicating a formerly chosen set of paths  *
*     *            and are not easements.               *
*     ***********************************************
*     * NLINK =number of links.
*     * NODE1,NODE2=node descriptors of links.
*     * REFLNK=link currently being considered.
*     * CHEKIN=array holding all links passed while 'checkin'.
*     * NUMPTH=number of paths possible paths found.
*     * PATHS =link numbers of possible paths.
*     * V,N1,N2,X1-X6=miscellaneous use.
*     * RESULT=indicates whether the path has already been tried.
*     * ENNODE=node trace should end on.

      INTEGER NLINK,NODE1(150),NODE2(150),REFLNK,CHEKIN(150),NUMPTH,
     &  PATHS(6),X1,X2,X3,X4,X5,X6,PCHIDX,POSCHN(10),CHNIDX,
     &  CHAINS(500,0:10),RESULT,FLAG,V

      REAL WROAD(150)

*     * Initialize variables.
      NUMPTH=0
      PATHS(1)=0
      PATHS(2)=0
      PATHS(3)=0
      PATHS(4)=0
      PATHS(5)=0
      PATHS(6)=0

      DO 10, L=1, NLINK
      ENNODE=0
      IF(PCHIDX.GT.1)THEN
      X1=NODE1(POSCHN(PCHIDX))
      X2=NODE2(POSCHN(PCHIDX))
      X3=NODE1(POSCHN(PCHIDX-1))
      X4=NODE2(POSCHN(PCHIDX-1))
      IF((X1.EQ.X3).OR.(X1.EQ.X4))ENNODE=X2
      IF((X2.EQ.X3).OR.(X2.EQ.X4))ENNODE=X1
      N1=ENNODE
      N2=ENNODE
      ENDIF

      IF(ENNODE.EQ.0)N1=NODE1(REFLNK)
      IF(ENNODE.EQ.0)N2=NODE2(REFLNK)
      X5=NODE1(L)
      X6=NODE2(L)
      IF((X5.EQ.N1).OR.(X5.EQ.N2).OR.(X6.EQ.N1).OR.(X6.EQ.N2))THEN
       IF(PCHIDX.LT.10)THEN
        PCHIDX=PCHIDX+1
        POSCHN(PCHIDX)=L
```

```
*     * Check for chain duplication.
      RESULT=0
      DO 20, L2=1, CHNIDX
      FLAG=0
      V=0
      DO WHILE((FLAG.EQ.0).AND.(V.LT.PCHIDX))
      V=V+1
      IF(POSCHN(V).NE.CHAINS(L2,V))FLAG=1
      ENDDO
      IF(FLAG.EQ.0) RESULT=1
20    CONTINUE

      IF((CHEKIN(L).EQ.0).AND.(WROAD(L).NE.0).AND.(RESULT.EQ.0))THEN
       NUMPTH=NUMPTH+1
       PATHS(NUMPTH)=L
      ENDIF
      POSCHN(PCHIDX)=0
      PCHIDX=PCHIDX-1
     ENDIF
    ENDIF

10 CONTINUE
   RETURN
   END
```

B.3.33 SUBROUTINE: PATHS4

```
      SUBROUTINE PATHS4(NLINK,NODE1,NODE2,REFLNK,CHEKIN,
     &  NUMPTH,PATHS,PCHIDX,POSCHN,CHNIDX,CHAINS)
*     ***********************************************
*     * This subroutine finds what paths are adjacent  *
*     * to a given link, and which, if chosen, wouldn't *
*     * cause a duplication in a formerly chosen set of *
*     *       paths.  Easements are considered here.    *
*     ***********************************************
*     * NLINK =number of links.
*     * NODE1,NODE2=node descriptors of links.
*     * REFLNK=link currently being considered.
*     * CHEKIN=array holding all links passed while 'checkin'.
*     * NUMPTH=number of paths possible paths found.
*     * PATHS =link numbers of possible paths.
*     * V,X1-X4 =miscellaneous use.
*     * RESULT=indicates whether the path has already been tried.

      INTEGER NLINK,NODE1(150),NODE2(150),REFLNK,CHEKIN(150),NUMPTH,
     &  PATHS(4),X1,X2,X3,PCHIDX,POSCHN(10),CHNIDX,CHAINS(500,0:10),
     &  RESULT,V

*     * Initialize variables.
      NUMPTH=0
      PATHS(1)=0
      PATHS(2)=0
      PATHS(3)=0
      PATHS(4)=0
      PATHS(5)=0
      PATHS(6)=0

      DO 10, L=1, NLINK
      X1=NODE2(REFLNK)
      X2=NODE1(L)
      IF((X1.EQ.X2).AND.(PCHIDX.LT.10))THEN
       PCHIDX=PCHIDX+1
       POSCHN(PCHIDX)=L

*      * Check for chain duplication.
       RESULT=0
       DO 20, L2=1, CHNIDX
       FLAG=0
       V=0
       DO WHILE((FLAG.EQ.0).AND.(V.LT.PCHIDX))
       V=V+1
       IF(POSCHN(V).NE.CHAINS(L2,V))FLAG=1
       ENDDO
       IF(FLAG.EQ.0) RESULT=1
20     CONTINUE

      IF((CHEKIN(L).EQ.0).AND.(RESULT.EQ.0))THEN
       NUMPTH=NUMPTH+1
       PATHS(NUMPTH)=L
      ENDIF
      POSCHN(PCHIDX)=0
      PCHIDX=PCHIDX-1
```

```
      ENDIF

   10 CONTINUE
      RETURN
      END
```

B.3.34 SUBROUTINE: PATHS5

```
      SUBROUTINE PATHS5(NLINK,NODE1,NODE2,REFNOD,NUMPTH,PATHS)
*       ***************************************************
*       *  This subroutine finds what links are adjacent  *
*       *                  to a given node.               *
*       ***************************************************
*       * NODE1,NODE2=nodes describing links.
*       * REFNOD=node currently being considered.
*       * NUMPTH=number of associations found.
*       * PATHS =link numbers of possible paths.
*       * X1,X2 =miscellaneous use.

      INTEGER NLINK,NODE1(150),NODE2(150),REFNOD,NUMPTH,PATHS(6),
        &  X1,X2,X3

*       * Initialize variables.
      NUMPTH=0
      PATHS(1)=0
      PATHS(2)=0
      PATHS(3)=0
      PATHS(4)=0
      PATHS(5)=0
      PATHS(6)=0

      DO 10, L=1, NLINK

      X1=NODE1(L)
      X2=NODE2(L)
      X3=0
      IF((X1.EQ.REFNOD).OR.(X2.EQ.REFNOD))THEN
      NUMPTH=NUMPTH+1
      IF(REFNOD.EQ.X1)X3=X2
      IF(REFNOD.EQ.X2)X3=X1
      PATHS(NUMPTH)=X3
      ENDIF

   10 CONTINUE
      RETURN
      END
```

B.3.35 SUBROUTINE: PATHS6

```
      SUBROUTINE PATHS6(REFLNK,CHEKIN,CSLEN,NUMPTH,PATHS,
     &  PCHIDX,POSCHN,CHNIDX,CHAINS)
*     ***********************************************
*     *  This subroutine finds what paths are adjacent  *
*     * to a given link and which would, if chosen, not *
*     *  be duplicating a formerly chosen set of paths  *
*     *                 and are not easements.          *
*     ***********************************************
*     * NLINK =number of links.
*     * NODE1,NODE2=node descriptors of links.
*     * REFLNK=link currently being considered.
*     * CHEKIN=array holding all links passed while 'checkin'.
*     * NUMPTH=number of paths possible paths found.
*     * PATHS =link numbers of possible paths.
*     * V,N1,N2,X-X7=miscellaneous use.
*     * RESULT=indicates whether the path has already been tried.
*     * ENNODE=node trace should end on.
*     * CSLEN =current shortest length.
*     * LLINK =link length.
*     * LENGTH=length of chain.

      INTEGER NODE1,NODE2,REFLNK,CHEKIN(150),NUMPTH,
     &  PATHS(6),X,X1,X2,X3,X4,X5,X6,X7,PCHIDX,POSCHN(50),CHNIDX,
     &  CHAINS(200,50),RESULT,ENNODE,NWMLNK,WLINKS,FLAG,V,N1,N2

      REAL WROAD,LLINK,LENGTH,CSLEN

      COMMON
     &      /AREA3/NLINK,NODE1(150),NODE2(150),LLINK(150),
     &      /AREA4/LROAD(150),WROAD(150),TROAD(150)
     &      /AREA6/NWMLNK,WLINKS(150),NWMGRP,TBTRYW(150,4)

*     * Initialize variables.
      NUMPTH=0
      PATHS(1)=0
      PATHS(2)=0
      PATHS(3)=0
      PATHS(4)=0
      PATHS(5)=0
      PATHS(6)=0

      X=1
      LENGTH=0.0
      DO WHILE((X.LE.PCHIDX).AND.(POSCHN(X).NE.0))
      LENGTH=LENGTH+LLINK(POSCHN(X))
      X=X+1
      ENDDO

      DO 10, L1=1, NWMLNK

*     * Find ENNODE.
      ENNODE=0
      IF(PCHIDX.GT.1)THEN
      X1=NODE1(POSCHN(PCHIDX))
      X2=NODE2(POSCHN(PCHIDX))
```

```
      X3=NODE1(POSCHN(PCHIDX-1))
      X4=NODE2(POSCHN(PCHIDX-1))
      IF((X1.EQ.X3).OR.(X1.EQ.X4))ENNODE=X2
      IF((X2.EQ.X3).OR.(X2.EQ.X4))ENNODE=X1
      N1=ENNODE
      N2=ENNODE
      ENDIF

      IF(ENNODE.EQ.0)N1=NODE1(REFLNK)
      IF(ENNODE.EQ.0)N2=NODE2(REFLNK)
      X5=WLINKS(L1)
      X6=NODE1(X5)
      X7=NODE2(X5)

      IF((X6.EQ.N1).OR.(X6.EQ.N2).OR.(X7.EQ.N1).OR.(X7.EQ.N2))THEN
       IF((PCHIDX.LT.50).AND.(LENGTH.LT.CSLEN))THEN
        PCHIDX=PCHIDX+1
        POSCHN(PCHIDX)=X5

*       * Check for chain duplication.
        RESULT=0
        DO 20, L2=1, CHNIDX
        FLAG=0
        V=0
        DO WHILE((FLAG.EQ.0).AND.(V.LT.PCHIDX))
        V=V+1
        IF(POSCHN(V).NE.CHAINS(L2,V))FLAG=1
        ENDDO
        IF(FLAG.EQ.0) RESULT=1
20      CONTINUE

        IF((CHEKIN(X5).EQ.0).AND.(WROAD(X5).NE.0).AND.(RESULT.EQ.0))THEN
         NUMPTH=NUMPTH+1
         PATHS(NUMPTH)=X5
        ENDIF
        POSCHN(PCHIDX)=0
        PCHIDX=PCHIDX-1

*       * Check to see if one path would result in a loop being formed.
        IF((X5.EQ.POSCHN(1)).AND.(PCHIDX.GT.2))THEN

*       * Find ENNODE.
         X1=NODE1(POSCHN(1))
         X2=NODE2(POSCHN(1))
         X3=NODE1(POSCHN(2))
         X4=NODE2(POSCHN(2))
         IF((X1.EQ.X3).OR.(X1.EQ.X4))ENNODE=X2
         IF((X2.EQ.X3).OR.(X2.EQ.X4))ENNODE=X1

*       * Check to see if the path is approaching from the right side.
*       * Update PATHS and NUMPTH if it is.
         X1=NODE1(REFLNK)
         X2=NODE2(REFLNK)
         IF(((X1.EQ.ENNODE).OR.(X2.EQ.ENNODE)).AND.(RESULT.EQ.0))THEN
          NUMPTH=NUMPTH+1
          PATHS(NUMPTH)=X5
         ENDIF
```

```
        ENDIF

      ENDIF
     ENDIF
   10 CONTINUE

     RETURN
     END
```

B.3.36 SUBROUTINE: PATHS7

```
      SUBROUTINE PATHS7(NODE1,NODE2,REFNOD,NSL,LINKS,PATHS)
*     ****************************************************
*     *  This subroutine finds what links are adjacent   *
*     *    to a given node and which are part of the     *
*     *                system being analyzed.            *
*     ****************************************************
*     * NODE1,NODE2=nodes describing links.
*     * REFNOD=node currently being considered.
*     * NSL   =number of system links.
*     * LINKS =links in system.
*     * NUMPTH=number of associations found.
*     * PATHS =link numbers of possible paths.
*     * X1,X2 =miscellaneous use.

      INTEGER NODE1(150),NODE2(150),REFNOD,NSL,LINKS(150),
     &  NUMPTH,PATHS(6),X1,X2

*     * Initialize variables.
      NUMPTH=0
      PATHS(1)=0
      PATHS(2)=0
      PATHS(3)=0
      PATHS(4)=0
      PATHS(5)=0
      PATHS(6)=0

      DO 10, L=1, NSL

      X1=NODE1(LINKS(L))
      X2=NODE2(LINKS(L))
      IF((X1.EQ.REFNOD).OR.(X2.EQ.REFNOD))THEN
      NUMPTH=NUMPTH+1
      PATHS(NUMPTH)=LINKS(L)
      ENDIF

   10 CONTINUE
      RETURN
      END
```

B.3.37 **SUBROUTINE: SAC**

```
      SUBROUTINE SAC(CNODE,I,ASAC,LEN)
*
*  ACENOD --- NODE( ALWAYS TYPED 2 ), INDICATED THE BEGINNING OF THE SAC
*  ACELIN --- THE LINK OF THE ACCESS ROAD
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,CNODE,ACENOD,ACELIN,ATTLIN

      REAL
     &  LLINK,LROAD,LENGTH,KM,IRAD,K,LEN

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      ANGLE(NODNUM),RCURB
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA26/ADJWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)

      ASAC=0.0
      LEN=0.0


*  DEFINE PI
      PI=ACOS(-1.0)

*  DETERMINE THE WIDTH OF THE CUL-DE-SAC ACCESS ROAD
      IF ( NODE1(I).EQ.CNODE )  THEN
              ACENOD=NODE2(I)
      ELSE
              ACENOD=NODE1(I)
      ENDIF

      DO 411 J=1,4

              IF ( ATTLIN(ACENOD,J).EQ.0 .OR. ATTLIN(ACENOD,J).EQ.I )
     &  GOTO411
              ACELIN=ATTLIN(ACENOD,J)

411           CONTINUE

*  WIDTH OF THE ACCESS ROAD
      WITH1=WROAD(ACELIN)

      IRAD=RADIUS(CNODE)-WROAD(I)
      CULSAC=0.5*PI*(RADIUS(CNODE)**2.0-IRAD**2.0)
      TRP=0.5*(2.0*RADIUS(CNODE)+WITH1)*LLINK(I)

*  DETERMINE THE RADIUS AND THE LENGHT OF THE END PART OF THE ISLAND
      IF ( LLINK(I).EQ.0.0 ) THEN
```

```
            PRAD=IRAD
            K=0.0
            GOTO 50

      ENDIF

      ANG=ATAN(IRAD/LLINK(I))*2.0

      IF ( ANG.GE.(PI/2.0) ) THEN

            PRAD=IRAD
            K=0.0

      ELSE

            PRAD=0.5
*
*           PRAD=0.5, IF DESIRED
            K=LLINK(I)-LLINK(I)*PRAD/IRAD

      ENDIF

   50 P1=0.5*PI*PRAD**2.0
      P2=0.5*K*(2.0*IRAD+2.0*PRAD)

      ASAC=CULSAC+TRP-P1-P2

      TV=IRAD+WROAD(I)/2.0
      LEN=TV*PI+2.0*(LLINK(I)**2.0+TV**2.0)**0.5

      END
```

B.3.38 SUBROUTINE: SSSDPTH

```
      SUBROUTINE SSSDPTH(CLINK,DPTH1,DPTH2)

*****************************************************************
*                                                               *
*                        SUBROUTINE SSSDPTH                     *
*                                                               *
*****************************************************************
*
*
*
*  XXXXXX -
*  ATTLIN - the links associate with the given node. ( NOTE: max. link can
*           attach to the given node is 4 )
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory.
*  SS     - subscript.
*  SSLINK - storm sewer elev. of a given link.
*  NODE1  - origin node of the current link.
*  NODE2  - dest. node of the current link.
*  DPTH1  - the depth of excavation at origin node.
*  DPTH2  - the depth of excavation at dest. node.
*  J      - loop counter
*
*
*
      PARAMETER
     %  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     %  ATTLIN,CNODE,CLINK,SS

      COMMON
     %      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     %      ANGLE(NODNUM)
     %      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     %      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     %      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     %      /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     %      /AREA32/SSLINK(NODNUM,4)
*
*
*
*  save the origin node
      CNODE=NODE1(CLINK)
*
*  find the correct subscript to store the current link's
*  storm sewer elev.
      DO 100 J=1,4
             IF ( ATTLIN(CNODE,J).EQ.CLINK )  THEN
                     SS=J
             ENDIF
100          CONTINUE
      SSLINK(CNODE,SS)=ELEV(CNODE)-DPTH1
```

```
*
*
*
*  save dest node
   CNODE=NODE2(CLINK)
*
*  find the correct subscript to store the current link's
*  storm sewer elev.
   DO 110 J=1,4
            IF ( ATTLIN(CNODE,J).EQ.CLINK )  THEN
                     SS=J
            ENDIF
110         CONTINUE
   SSLINK(CNODE,SS)=ELEV(CNODE)-DPTH2
*
*
*
   RETURN
   END
```

B.3.39 SUBROUTINE: SWTABLE

```
      SUBROUTINE SWTABLE(COL,DIA,UCOST,TCOST,PIPLEN)


*****************************************************************
*                                                               *
*                    SEWER/WATERMAIN SYSTEM TABLE               *
*                                                               *
*****************************************************************
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER XSEC,XSPA,YSEC,COL,YTOTAL,X,SPACE,WUNIT1

      REAL
     &      LLINK,DIA(LINNUM),UCOST(LINNUM),TCOST(LINNUM)

      CHARACTER*132 LINE,LINE1,LINE2,LINE3,LINE4

      COMMON
     &      /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &      RCURB
     &      /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &      UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &      /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &      /AREA39/LINE,LINE1,LINE2,LINE3,LINE4
*
*
*
*     create the table
      YSEC=5
      XSEC=4
      XSPA=30
      CALL CLINE(XSPA,XSEC,COL,0,0)
      MCOL=XSPA*XSEC
*
      LENGTH=LEN('PIPE LENGTH')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+2
      LINE3(IFRONT:IFRONT+LENGTH)='PIPE LENGTH'
*
      LENGTH=LEN('LINK')
      IFRONT=INT((5-LENGTH)/2.0)+2
      LINE3(IFRONT:IFRONT+LENGTH)='LINK'
*
      LENGTH=LEN('DIAMETER')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA+SPACE
      LINE3(IFRONT:IFRONT+LENGTH)='DIAMETER'
*
      LENGTH=LEN('UNIT COST')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*2+SPACE
      LINE3(IFRONT:IFRONT+LENGTH)='UNIT COST'
*
```

```
      LENGTH=LEN('TOTAL COST')
      IFRONT=INT((XSPA-LENGTH)/2.0)+COL+XSPA*3+SPACE
      LINE3(IFRONT:IFRONT+LENGTH)='TOTAL COST'
*
      LINE3(COL+XSPA:COL+XSPA)='!'
      LINE4(COL+XSPA*3:COL+XSPA*3)='!'
*
      WRITE (WUNIT1,400) LINE
      WRITE (WUNIT1,400) LINE3
      WRITE (WUNIT1,400) LINE1
*
      PIPLEN=0.0
      YTOTAL=YSEC+1
      DO 100 L=1,NLINK
              IF ( L.EQ.YTOTAL )  THEN
                WRITE (WUNIT1,400) LINE1
                YTOTAL=YTOTAL+YSEC
              ENDIF
              IF ( DIA(L).GT.0.0 )  THEN
                PIPE=LLINK(L)
                PIPLEN=PIPLEN+LLINK(L)
              ELSE
                PIPE=0.0
              ENDIF
              TCOST(L)=UCOST(L)*PIPE
*
              WRITE (WUNIT1,500) L,PIPE,DIA(L),UCOST(L),TCOST(L)
100           CONTINUE
*
      WRITE (WUNIT1,400) LINE
*
*
*
400           FORMAT (1X,A132)
500           FORMAT (1X,'!',1X,I3,1X,'!',8X,F10.2,'m',10X,'!',8X,F10.2
     &        ,'mm',9X,'!',2(5X,'$',3X,F10.2,10X,'!'))
*
*
*
      RETURN
      END
```

B.3.40 SUBROUTINE: TAGPIT

```
      SUBROUTINE TAGPIT(NLINK,NODE1,NODE2,ELEV,REFNOD,PITFLG)
*       ***************************************************
*       * This routine sets the flag if a node is in a pit. *
*       ***************************************************
*       * REFNOD=the node being analyzed.
*       * PITFLG=pit status flag.
*       * NUMPTH=number of possible paths.
*       * PATHS =a list of ID's of possible paths.

      INTEGER NLINK,NODE1(100),NODE2(100),REFNOD,
     &  PITFLG,NUMPTH,PATHS(6)

      REAL ELEV(100)

      CALL PATHS5(NLINK,NODE1,NODE2,REFNOD,NUMPTH,PATHS)
      PITFLG=0
      DO 10, L=1, 4
      IF(PATHS(L).NE.0)THEN
       IF(ELEV(REFNOD).LE.ELEV(PATHS(L)))NUMPTH=NUMPTH-1
       IF(NUMPTH.EQ.0)PITFLG=1
          ENDIF
   10 CONTINUE

      END
```

B.3.41 SUBROUTINE: THREEWAY

```
      SUBROUTINE THREEWAY(ORIGIN,DEST,BRANCH,LAST,EXRLEN)

*****************************************************************
*                                                               *
*                          THREEWAY CHECKER                     *
*                                                               *
*****************************************************************
*
*
*
*  ATTLIN - the links associate with the current node. ( NOTE: max. link ca
*           attach to the current node is 4 )
*  CLINK  - current link in the memory.
*  CNODE  - current node in the memory.
*  NTYPE  - node type.
*  ORIGIN - the origin node of the sequence of nodes comprising a street
*           ( one street may be composed of several links )
*  DEST   - the dest node of the sequence of nodes comprising a street
*           ( one street may be composed of several links )
*  BRANCH - the branch of the path
*  PATH   - the links between two intersections or intersection/dead
*           end/cul-de-sac
*  TROAD  - type of road/street
*  CORWD  - corresponding road width of the given link.
*  DIRECT - a flag indicates the origin/dest node for a given link:
*  LAST   - the last link between two intersections or intersection/dead
*           end/cul-de-sac of the current branch.
*  EXRLEN - extra street length
*  LINK   - store the current link
*  RATIO  - the street length ratio
*  LOPCON - loop controller
*  EXIT   - exit node.
*  K      - loop counter
*
*
*
      PARAMETER
     &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
     &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,ATTLIN,TROAD,
     &     CNODE,CLINK,PATH,BRANCH,ORIGIN,DEST,DIRECT,EXIT

      COMMON
     &     /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
     &     ANGLE(NODNUM)
     &     /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
     &     UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
     &     /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
     &     /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
     &     /AREA26/CORWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
     &     /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)
     &     /AREA28/STRLEN(NODNUM,NODNUM,4),PATH(NODNUM,4,NODNUM)
*
```

```
*
*
*  set the extra street length ( width of the road/street ) to zero
   EXRLEN=0.0
*  check the street condition
*  look for three way intersection
*  determine the value of the loop controller
   IF ( NTYPE(ORIGIN).EQ.3 .AND. NTYPE(DEST).EQ.3 )  THEN
           LOPCON=2
           CNODE=ORIGIN
   ELSE
           IF ( NTYPE(ORIGIN).EQ.3 .OR. NTYPE(DEST).EQ.3 )  THEN
             LOPCON=1
             IF ( NTYPE(ORIGIN).EQ.3 )  THEN
               CNODE=ORIGIN
             ELSE
               CNODE=DEST
             ENDIF
           ELSE
             LOPCON=0
           ENDIF
   ENDIF
*
*  locate the three way intersection
*
   DO 100 K=1,LOPCON
*  determine the street length ratio
           IF ( K.GE.2)  THEN
             CNODE=DEST
             RATIO=0.5
           ELSE
             RATIO=1.0
           ENDIF
           IF ( LAST.EQ.1 )  THEN
             CLINK=PATH(ORIGIN,BRANCH,1)
             IF ( (NTYPE(ORIGIN).EQ.3.OR.NTYPE(ORIGIN).EQ.4) .AND.
     &                (NTYPE(DEST).EQ.3.OR.NTYPE(DEST).EQ.4) )  THEN
               IF ( TROAD(CLINK).EQ.1 )  THEN
                 RATIO=0.5
               ENDIF
             ENDIF
           ENDIF
*
*
           IF ( ORIGIN.NE.DEST )  THEN
*  check the origin link
             CLINK=PATH(ORIGIN,BRANCH,1)
             DO 110 J=1,4
               IF ( CLINK.EQ.ATTLIN(CNODE,J) )  THEN
                 LINK=CLINK
               ENDIF
110                  CONTINUE
*
*  check the dest link
             CLINK=PATH(ORIGIN,BRANCH,LAST)
             DO 120 J=1,4
               IF ( CLINK.EQ.ATTLIN(CNODE,J) )  THEN
```

```
                LINK=CLINK
              ENDIF
120                 CONTINUE
          ELSE
            IF ( K.EQ.1 )  THEN
              LINK=PATH(ORIGIN,BRANCH,1)
            ELSE
              LINK=PATH(ORIGIN,BRANCH,LAST)
            ENDIF
          ENDIF
*
*
*  calculate the extra street length for placing street lights
          IF ( NODE1(LINK).EQ.CNODE )  THEN
            DIRECT=1
          ELSE
            DIRECT=2
          ENDIF
*
*  elimiate the subdivision's exits
          IF ( EXIT(CNODE).EQ.0 )  THEN
                EXRLEN=EXRLEN+RATIO*CORWD(LINK,DIRECT)
          ENDIF
*
*
*
   WRITE (088,200) ORIGIN,DEST
   WRITE (088,205) CNODE,LINK
   IF ( DIRECT.EQ.1 )  THEN
          WRITE(088,210) NODE1(LINK)
   ELSE
          WRITE(088,210) NODE2(LINK)
   ENDIF
   WRITE (088,220) RATIO,EXRLEN

200       FORMAT (1X,' ORIGIN NODE :',3X,I3,3X,' DEST NODE :',3X,I3)
205       FORMAT (1X,'CURRENT NODE :',3X,I3,3X,' LINK      :',3X,I3)
210       FORMAT (1X,'   FROM NODE :',3X,I3)
220       FORMAT (1X,'       RATIO :',3X,F9.3,3X,'EXTRA :',3X,F9.3,//)

*
100       CONTINUE
*
*
*
   RETURN
   END
```

B.3.42 SUBROUTINE: TUBE

```
      SUBROUTINE TUBE(CNODE,I,ASAC,LEN)


      PARAMETER
        &  NODNUM=100,LINNUM=150,INTNUM=10

      INTEGER
        &  RUNIT1,RUNIT2,WUNIT1,WUNIT2,CNODE

      REAL
        &  LLINK,LROAD,LENGTH,KM,IRAD,ISLAND,LEN

      COMMON
        &       /AREA2/NNODE,NTYPE(NODNUM),ELEV(NODNUM),RADIUS(NODNUM),
        &       ANGLE(NODNUM),RCURB
        &       /AREA3/NLINK,NODE1(LINNUM),NODE2(LINNUM),LLINK(LINNUM),
        &       UNITS(LINNUM),MULTI(LINNUM),AREA(LINNUM)
        &       /AREA4/LROAD(LINNUM),WROAD(LINNUM),TROAD(LINNUM)
        &       /AREA21/RUNIT1,RUNIT2,WUNIT1,WUNIT2
        &       /AREA26/ADJWD(LINNUM,2),AINTS(NODNUM),ALINK(LINNUM)
        &       /AREA27/ATTLIN(NODNUM,4),EXIT(NODNUM)

      ASAC=0.0
      LEN=0.0


*     DEFINE PI
      PI=ACOS(-1.0)

*     MIN. RADIUS
      IRAD=0.5

*     ROAD LENGTH
      TV=LLINK(I)-(ADJWD(I,1)+ADJWD(I,2))/2.0

*     LENGHT OF THE ISLAND
      D=TV-2.0*(RADIUS(CNODE)-IRAD)

*     SIZE OF THE ISLAND
      ISLAND=IRAD**2.0*PI+D*IRAD*2.0

*     DETERMINE THE AREA

      IF ( RADIUS(CNODE).GE.WROAD(I) )  THEN

              ASAC=0.5*RADIUS(CNODE)**2.0*PI+TV*WROAD(I)-
        &ISLAND

      ELSE

              ASAC=0.5*RADIUS(CNODE)**2.0*PI+TV*2*(WROAD(I)+IRAD)-
        &ISLAND

      ENDIF
```

```
      LEN=TV*2.0+RADIUS(CNODE)*PI

      RETURN
      END
```

B.4 FUNCTIONS USED BY RACAM

B.4.1 SUBROUTINE: ARITHA

```
      FUNCTION ARITHA (BASE,GRAD,INTRST,N)

      REAL ARITHA,BASE,GRAD,INTRST

      INTEGER N

      RATE=INTRST/100.0
      W=(1.0+RATE)**N
*
      ARITHA=BASE+GRAD*(1.0/RATE-N/(W-1.0))
*
*
*
      END
```

B.4.2 SUBROUTINE: ARITHF

```
      FUNCTION ARITHF (BASE,GRAD,INTRST,N)

      REAL ARITHF,BASE,GRAD,INTRST

      INTEGER N

      RATE=INTRST/100.0
      W=(1.0+RATE)**N
*
*     find the lump sum
      ARITHF=BASE*(W-1.0)/RATE
*
*     add ARITHF to the gradient series
      ARITHF=GRAD/RATE*((W-1.0)/RATE-N)+ARITHF
*
*
*
      END
```

B.4.3 SUBROUTINE: ARITHP

```
      FUNCTION ARITHP (BASE,GRAD,INTRST,N)

      REAL ARITHP,BASE,GRAD,INTRST

      INTEGER N

      RATE=INTRST/100.0
      W=(1.0+RATE)**N
*
*     find the lump sum
      ARITHP=BASE*(W-1.0)/(RATE*W)
*
*     add ARITHP to the gradient series
      ARITHP=GRAD/RATE*((W-1.0)/(RATE*W)-N/W)+ARITHP
*
*
*
      END
```

B.4.4 SUBROUTINE: UNFORM

```
FUNCTION UNFORM(INTRST,N,PWORTH)

REAL UNFORM,INTRST,PWORTH

INTEGER N
RATE=INTRST/100.0
W=(1.0+RATE)**N
UNFORM=PWORTH*(RATE*W/(W-1.0))

END
```

B.4.5 SUBROUTINE: UNFORM1

```
FUNCTION UNFORM1(INTRST,N,AWORTH)

REAL UNFORM,INTRST,AWORTH

INTEGER N

RATE=INTRST/100.0
W=(1.0+RATE)**N
UNFORM1=AWORTH*((W-1.0)/(RATE*W))

END
```

B.4.6 SUBROUTINE: UNFORM2

```
FUNCTION UNFORM2(INTRST,N,PWORTH)

REAL UNFORM,INTRST,PWORTH

INTEGER N

RATE=INTRST/100.0
UNFORM2=PWORTH*(1.0+RATE)**N

END
```

APPENDIX C

SAMPLE DATA FILES

This appendix contains samples of the two data files required by RACAM. These samples files use data compiled from a study of Rykmans Neighbourhood, a subdivision in Hamilton, Ontario. All data in the data files, except the accour data in the economic data file, is therefore actual data. The account data, including account names, numbers and percentages, is fictional data used only tr indicate the format of this portion of the data file.

The first sample file consists of the economic data, including unit costs, RACAM constants and account information. The second sample file consists of link and node data, including link and node characteristics and subdivision areas. Both types of data files are required for RACAM execution.

APPENDIX C SAMPLE DATA FILES

C.1 ECONOMIC DATA

```
       5.0          1
       5.0         10
       5.0         20
       5.0         50
       5.0        100
      13.5         10
      13.5         20
      13.5         50
      13.5        100
       -1.         -1
  150.00
 1500.00
50000.00
50000.00
   30.00
   25.00
   45.00
   20.00
   15.00
   15.00
  140.00  165.00  180.00  190.00  420.00   00.00
  270.00  310.00  324.00  340.00  356.00  372.00  390.00  410.00  000.00
  272.00  314.00  328.00  344.00  360.00  378.00  396.00  416.00  000.00
  312.00  358.00  372.00  388.00  406.00  424.00  442.00  462.00  000.00
  314.00  362.00  378.00  394.00  412.00  432.00  450.00  472.00  000.00
   00.00   00.00   00.00   00.00   00.00   00.00   00.00   00.00  000.00
  404.00  454.00  468.00  484.00  502.00  520.00  538.00  558.00  000.00
   10.00  458.00  474.00  490.00  508.00  528.00  546.00  568.00  000.00
  414.00  462.00  480.00  496.00  516.00  534.00  556.00  576.00  000.00
  594.00  646.00  664.00  682.00  700.00  720.00  742.00  764.00  000.00
  596.00  650.00  668.00  688.00  708.00  748.00  750.00  774.00  000.00
  600.00  660.00  680.00  700.00  722.00  744.00  768.00  792.00  000.00
  766.00  832.00  854.00  876.00  898.00  922.00  948.00  974.00  000.00
  772.00  842.00  864.00  888.00  912.00  938.00  964.00  992.00  000.00
  776.00  852.00  876.00  900.00  926.00  954.00  982.00 1010.00  000.00
  839.00  921.00  945.00  973.00  999.00 1029.00 1057.00 1089.00  000.00
  843.00  931.00  957.00  985.00 1015.00 1045.00 1075.00 1107.00  000.00
  849.00  939.00  969.00  997.00 1029.00 1059.00 1093.00 1125.00  000.00
 1026.00 1122.00 1152.00 1184.00 1216.00 1248.00 1282.00 1318.00  000.00
  000.00  000.00  000.00  000.00  000.00  000.00  000.00  000.00  000.00
  960.0
  720.00
  900.00
  558.00
  404.00   454.00   468.00   484.00   502.00   520.00   538.00   558.00  598.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
 1284.00
   43.00
 1306.00
 1124.00
```

```
 1140.00
 1980.00
  425.00
   17.34
  347.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
    0.00
123-456-7897          Aloysius & Sons, Rotorooters
234-345-4561               Sammy's Cement Company
567-678-6789                    City of Cucamonga
132-354-4657                 Clancy's Construction
756-143-6472            Bonafide Buildings Limited
          -1                                    -1
          2
          2        25.
          3        75.
          2
          1        51.
          3        49.
          4
          1        20.
          2        25.
          3        25.
          4        30.
          3
          1        35.
          3        25.
          5        40.
          2
          2        69.
          3        31.
          1
          4       100.
          2
          4        50.
          5        50.
          3
          2        10.
          3        20.
          5        70.
          2
          1        98.
          3         2.
          5
          1        20.
```

```
   2        30.
   3        40.
   4         5.
   5         5.
   2
   1        10.
   5        90.
   1
   2       100.
   4
   1        30.
   2        25.
   3        20.
   5        25.
   2
   2        50.
   3        50.
   1
   5       100.
   3
   3        15.
   4        29.
   5        56.
   2
   2        10.
   4        90.
   4
   1        19.
   3         1.
   4        32.
   5        48.
   1
   3       100.
   5
   1        20.
   2         4.
   3        26.
   4        25.
   5        25.
   1
   5       100.
  11.       401.        8.        4
  13.       370.        6.        6
  10.       490.        3.        8
  15.       500.        7.        5
  12.       476.        5.        7
  11.       401.        8.        4
  13.       370.        6.        6
  10.       490.        3.        8
  15.       500.        7.        5
  12.       476.        5.        7
  11.       401.        8.        4
  13.       370.        6.        6
  10.       490.        3.        8
  15.       500.        7.        5
  12.       476.        5.        7
  11.       401.        8.        4
```

```
    13.      370.      6.      6
    10.      490.      3.      8
    15.      500.      7.      5
    12.      476.      5.      7
    11.      401.      8.      4
    13.      370.      6.      6
    10.      490.      3.      8
    15.      500.      7.      5
    12.      476.      5.      7
    .85       11.   6.34       9
     1.       13.   4.56      10
     1.       15.   3.99       7
    .87       11.   7.91       8
    .95       12.   5.67      11
    .85       11.   6.34       9
     1.       13.   4.56      10
     1.       15.   3.99       7
    .87       11.   7.91       8
    .95       12.   5.67      11
    .85       11.   6.34       9
     1.       13.   4.56      10
     1.       15.   3.99       7
    .87       11.   7.91       8
    .95       12.   5.67      11
    .85       11.   6.34       9
     1.       13.   4.56      10
     1.       15.   3.99       7
    .87       11.   7.91       8
    .95       12.   5.67      11
    .85       11.   6.34       9
     1.       13.   4.56      10
     1.       15.   3.99       7
    .87       11.   7.91       8
    .95       12.   5.67      11
```

C.2 LINK AND NODE DATA

-3	17.93	0.00
3	20.05	0.00
1	16.41	15.85
5	13.51	0.00
5	10.92	0.00
-5	10.92	0.00
3	20.22	0.00
2	16.87	0.00
2	19.46	0.00
3	19.46	0.00
1	17.32	13.41
3	17.32	0.00
2	21.28	0.00
3	20.98	0.00
2	19.76	0.00
1	19.46	14.63
4	17.17	0.00
-3	13.36	0.00
-3	26.16	0.00
3	24.49	0.00
3	24.79	0.00
3	21.59	0.00
2	21.13	20.73
2	18.69	0.00
2	20.83	0.00
2	19.46	0.00
3	19.76	0.00
1	18.85	15.85
3	16.41	0.00
-3	9.55	0.00
3	13.97	0.00
2	14.20	0.00
3	14.43	0.00
2	17.02	0.00
2	15.65	0.00
1	17.17	14.63
3	17.93	0.00
1	18.69	14.63
3	18.39	0.00
2	24.79	0.00
3	21.89	0.00
-3	25.55	0.00
3	20.98	0.00
3	21.50	0.00
1	24.79	15.85
3	23.27	0.00
2	21.44	0.00
1	22.66	15.85
-3	20.22	0.00
4	20.98	0.00
3	23.11	0.00
1	18.24	15.85
2	18.85	0.00
3	16.26	0.00
3	17.17	0.00

2	15.49	0.00				
3	16.41	0.00				
2	13.36	0.00				
-5	13.66	0.00				
2	15.80	0.00				
-3	14.43	0.00				
99	99.00	99.00				
2	1	126.49	0.00	0.00	0.56	0.00
2	3	120.40	14.00	0.00	0.82	257.55
3	4	161.54	0.00	0.00	1.28	0.00
4	5	121.92	0.00	0.00	0.66	0.00
5	6	38.10	0.00	0.00	0.16	0.00
8	4	56.39	0.00	0.00	0.13	0.00
7	2	89.92	0.00	0.00	0.32	0.00
7	8	246.89	32.00	0.00	1.26	464.82
12	8	74.68	0.00	10.00	0.90	0.00
7	10	60.96	0.00	0.00	0.19	0.00
14	9	76.20	0.00	0.00	1.07	0.00
9	10	246.89	86.00	35.00	2.13	323.09
10	11	112.78	0.00	0.00	0.31	0.00
11	12	167.64	30.00	0.00	0.95	324.90
12	17	80.77	0.00	30.00	0.83	0.00
16	17	167.64	32.00	0.00	0.91	324.90
17	18	120.40	9.00	37.00	1.18	103.63
17	29	80.77	0.00	0.00	0.89	0.00
15	10	71.63	0.00	0.00	0.80	0.00
14	15	160.02	34.00	0.00	1.13	472.44
13	14	54.86	4.00	25.00	0.44	0.00
20	13	48.77	2.00	13.00	0.28	24.38
19	20	134.11	0.00	0.00	0.63	0.00
21	20	45.72	0.00	0.00	0.18	0.00
21	40	70.10	15.00	0.00	0.41	161.54
40	41	38.10	3.00	0.00	0.46	0.00
21	22	83.82	9.00	0.00	0.39	143.26
22	23	73.15	0.00	0.00	0.63	92.66
23	24	123.44	27.00	0.00	0.69	251.46
25	24	80.77	7.00	0.00	0.71	97.54
22	25	152.40	25.00	0.00	0.82	381.00
24	34	83.82	0.00	0.00	1.92	0.00
41	39	99.06	7.00	0.00	0.58	89.92
38	39	21.34	6.00	0.00	0.22	85.34
39	37	82.30	7.00	0.00	0.39	76.20
37	36	59.44	6.00	0.00	0.63	77.72
36	35	85.34	0.00	0.00	0.22	0.00
35	33	169.16	19.00	0.00	0.83	239.27
34	35	59.44	6.00	0.00	0.25	48.77
26	34	57.91	8.00	0.00	0.29	124.97
27	26	82.30	0.00	0.00	0.35	36.58
27	29	179.83	27.00	0.00	0.99	329.18
29	31	82.30	0.00	26.00	0.68	0.00
31	30	89.92	0.00	125.00	1.77	0.00
32	31	83.82	7.00	23.00	0.73	89.61
33	32	80.77	2.00	0.00	1.15	41.45
28	27	57.91	14.00	0.00	0.50	121.92
57	33	54.86	0.00	0.00	0.55	0.00
55	54	76.20	0.00	0.00	0.57	216.41
54	61	196.60	33.00	0.00	1.05	416.05

58	59	42.67	0.00	0.00	0.00	0.00
56	57	39.62	1.00	0.00	0.15	0.00
51	56	277.37	26.00	0.00	1.36	399.29
46	51	135.64	19.00	0.00	1.24	256.03
45	46	64.01	13.00	0.00	0.65	201.17
46	44	100.58	0.00	20.00	1.14	0.00
44	47	164.59	23.00	0.00	0.80	277.37
47	37	85.34	16.00	0.00	0.83	187.45
43	44	57.91	0.00	20.00	0.47	0.00
42	43	170.69	0.00	0.00	0.97	0.00
41	43	94.49	6.00	24.00	0.59	89.92
51	50	91.44	0.00	0.00	0.60	0.00
48	50	67.06	13.00	0.00	0.45	198.12
50	49	121.92	11.00	0.00	0.66	156.97
50	55	175.26	29.00	0.00	0.85	153.92
52	53	33.53	11.00	0.00	0.36	128.02
53	54	86.87	12.00	0.00	0.46	121.92
55	60	176.78	31.00	0.00	1.02	513.59
60	58	103.63	13.00	0.00	2.74	115.82
57	58	114.30	18.00	0.00	1.04	216.41
-1	-1	-1.00	-1.00	-1.00	-1.00	-1.00

135.64	26.00	1
240.79	10.00	2
0.00	0.00	2
0.00	0.00	2
0.00	0.00	2
0.00	0.00	2
82.30	20.00	1
179.83	20.00	1
74.68	20.00	1
60.96	20.00	1
64.01	20.00	2
245.36	20.00	2
0.00	0.00	2
347.47	10.00	2
80.77	20.00	1
347.47	10.00	2
112.78	20.00	1
82.30	20.00	1
71.63	20.00	1
220.98	20.00	1
56.39	20.00	1
48.77	20.00	1
121.92	26.00	1
45.72	20.00	1
68.58	20.00	1
38.10	20.00	1
74.68	20.00	2
65.53	20.00	2
124.97	20.00	2
82.30	20.00	2
150.88	20.00	2
0.00	0.00	2
88.39	20.00	2
53.34	15.00	2
82.30	20.00	2
149.35	10.00	2

0.00	0.00	2	
161.54	20.00	2	
59.44	20.00	2	
57.91	20.00	2	
80.77	20.00	2	
173.74	20.00	2	
73.15	20.00	1	
100.58	26.00	1	
73.15	20.00	1	
82.30	20.00	1	
124.97	10.00	2	
54.86	20.00	1	
60.96	20.00	2	
196.60	20.00	2	
0.00	0.00	2	
38.10	20.00	1	
278.89	20.00	1	
135.64	20.00	1	
128.02	10.00	2	
100.58	20.00	1	
153.92	20.00	2	
77.72	20.00	2	
57.91	20.00	1	
163.07	20.00	1	
94.49	20.00	1	
82.30	26.00	1	
128.02	10.00	2	
120.40	26.00	1	
163.07	20.00	2	
88.39	10.00	2	
88.39	20.00	2	
178.31	20.00	2	
103.63	20.00	2	
106.68	20.00	2	
1.5	140.2	106.7	0.0
1.2	6.1	262.1	0.0
0.0	0.0	0.0	9.1
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
1.2	91.4	67.1	0.0
1.2	6.1	472.4	0.0
1.2	3.0	134.1	0.0
1.2	50.3	48.8	0.0
1.2	6.1	125.0	0.0
1.2	103.6	373.4	0.0
0.0	0.0	0.0	0.0
1.2	6.1	353.6	9.1
1.2	9.1	125.0	0.0
1.2	6.1	353.6	9.1
1.5	6.1	217.9	0.0
1.2	9.1	125.0	0.0
1.2	88.4	54.9	0.0
1.2	24.4	410.0	0.0
1.2	3.0	105.2	0.0
1.2	3.0	82.3	0.0
1.5	6.1	243.8	0.0

1.2	6.1	61.0	0.0
1.2	3.0	126.5	0.0
1.2	3.0	59.4	0.0
1.2	9.1	135.6	0.0
1.2	6.1	136.5	0.0
1.2	3.0	245.4	0.0
1.2	6.1	158.5	0.0
1.2	6.1	286.5	0.0
0.0	0.0	0.0	0.0
1.2	9.1	153.9	0.0
1.2	6.1	64.0	7.6
1.2	6.1	131.1	0.0
1.2	3.0	139.5	9.1
0.0	0.0	0.0	0.0
1.2	9.1	306.3	0.0
1.2	3.0	120.4	0.0
1.2	3.0	109.7	0.0
1.2	91.4	59.4	0.0
1.2	15.2	164.6	0.0
1.2	9.1	128.0	0.0
1.5	3.0	169.2	0.0
1.2	3.0	155.5	0.0
1.2	3.0	149.4	0.0
1.2	6.1	134.3	9.1
1.2	6.1	82.3	0.0
1.2	12.2	109.7	0.0
1.2	3.0	376.4	0.0
0.0	0.0	0.0	0.0
1.2	3.0	67.1	0.0
1.2	107.5	356.6	0.0
1.2	6.1	253.0	0.0
1.2	6.1	163.1	9.1
1.2	6.1	179.8	0.0
1.2	9.1	297.2	0.0
1.2	9.1	149.4	0.0
1.2	6.1	56.4	0.0
1.5	6.1	321.6	0.0
1.2	6.1	163.1	0.0
1.2	12.2	128.0	0.0
1.2	6.1	157.0	9.1
1.5	6.1	214.9	0.0
1.2	9.1	306.3	0.0
1.2	3.0	112.8	9.1
1.2	6.1	160.0	0.0
1.2	6.1	341.4	0.0
1.2	6.1	199.6	0.0
1.2	9.1	211.8	0.0
11	1.0	0.0	0.0
18	1.0	0.0	0.0
19	1.0	0.0	0.0
-1	-1.0	-1.0	-1.0
7.37			
0.28			
7.5			
12.0			
13.1			
1.9			

76.9
-1

APPENDIX D

SAMPLE OUTPUT FILE

This appendix contains a sample RACAM output data file. The input da used for this RACAM run was obtained for Rykman's Neighbourhood, Ontion "C" Hamilton, Ontario. Some of the economic data used for this output is fictional, intended only to demonstrate RACAM operation (See Appendix C).

APPENDIX D SAMPLE OUTPUT FILE

D.1 UNIT PRICES

UNIT PRICES
==========

DESCRIPTION	UNITS	UNIT PRICE
STREET SIGN	$/INTERSECTION	150.00
STREET LIGHT	$/LIGHT	1500.00
COLLECTOR STREET	$/SQ M	30.00
LOCAL STREET	$/SQ M	25.00
SIDEWALK	$/SQ M	45.00
BARRIER CURB & GUTTER	$/M	20.00
ROLLED CURB & GUTTER	$/M	15.00
BLVD CURB & GUTTER	$/M	15.00
PARK DEVELOPMENT	$/HA	50000.00
BUFFER DEVELOPMENT	$/HA	50000.00
POWER (SINGLE FAMILY)	$/UNIT	0.00
(MULTI FAMILY)	$/UNIT	0.00
GAS (SINGLE FAMILY)	$/UNIT	0.00
(MULTI FAMILY)	$/UNIT	0.00
PHONE (SINGLE FAMILY)	$/UNIT	0.00
(MULTI FAMILY)	$/UNIT	0.00
STREET SWEEPING	$/KM/YR	1284.00
WINTER ROAD	$/KM/YR	2043.00
ASPHALT MAINTENANCE	$/KM/YR	1606.00
CONCRETE MAINTENANCE	$/KM/YR	1124.00
SEWER MAINTENANCE	$/KM/YR	1140.00
WATER MAINTENANCE	$/KM/YR	1980.00
SOLID WASTE COLLECTION	$/HA/YR	425.00
STREET LIGHT MAINTENANCE	$/LIGHT/YR	17.34
PARK & BUFFER MAINTENANCE	$/HA/YR	347.00
CATCHBASIN		
BASE, FRAME, COVER, ETC.	$/CATCHBASIN	900.00
SHAFT	$/M-depth	558.00
LEAD DEPTH INDEX = 1	$/M	404.00
DEPTH INDEX = 2	$/M	454.00
DEPTH INDEX = 3	$/M	468.00
DEPTH INDEX = 4	$/M	484.00
DEPTH INDEX = 5	$/M	502.00
DEPTH INDEX = 6	$/M	520.00
DEPTH INDEX = 7	$/M	538.00
DEPTH INDEX = 8	$/M	558.00
DEPTH INDEX = 9	$/M	598.00

MANHOLE

BASE, FRAME, COVER, ETC.	$/MANHOLE	960.00
SHAFT	$/M-depth	720.00

WATERMAIN @ 2.44 M OF COVER

DIAMETER INDEX = 1	$/M	140.00
DIAMETER INDEX = 2	$/M	165.00
DIAMETER INDEX = 3	$/M	180.00
DIAMETER INDEX = 4	$/M	190.00
DIAMETER INDEX = 5	$/M	420.00
DIAMETER INDEX = 6	$/M	0.00

DOMESTIC SEWER ($/M)

	DEPTH INDEX								
	1	2	3	4	5	6	7	8	9
DIAMETER INDEX = 1	270.00	310.00	324.00	340.00	356.00	372.00	390.00	410.00	0.00
DIAMETER INDEX = 2	272.00	314.00	328.00	344.00	360.00	378.00	396.00	416.00	0.00
DIAMETER INDEX = 3	312.00	358.00	372.00	388.00	406.00	424.00	442.00	462.00	0.00
DIAMETER INDEX = 4	314.00	362.00	378.00	394.00	412.00	432.00	450.00	472.00	0.00
DIAMETER INDEX = 5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

STORM SEWER ($/M)

	DEPTH INDEX								
	1	2	3	4	5	6	7	8	9
DIAMETER INDEX = 1	404.00	454.00	468.00	484.00	502.00	520.00	538.00	558.00	0.00
DIAMETER INDEX = 2	410.00	458.00	474.00	490.00	508.00	528.00	546.00	568.00	0.00
DIAMETER INDEX = 3	414.00	462.00	480.00	496.00	516.00	534.00	556.00	576.00	0.00
DIAMETER INDEX = 4	594.00	646.00	664.00	682.00	700.00	720.00	742.00	764.00	0.00
DIAMETER INDEX = 5	596.00	650.00	668.00	688.00	708.00	748.00	750.00	774.00	0.00
DIAMETER INDEX = 6	600.00	660.00	680.00	700.00	722.00	744.00	768.00	792.00	0.00
DIAMETER INDEX = 7	766.00	832.00	854.00	876.00	898.00	922.00	948.00	974.00	0.00
DIAMETER INDEX = 8	772.00	842.00	864.00	888.00	912.00	938.00	964.00	992.00	0.00
DIAMETER INDEX = 9	776.00	852.00	876.00	900.00	926.00	954.00	982.00	1010.00	0.00
DIAMETER INDEX = 10	839.00	921.00	945.00	973.00	999.00	1029.00	1057.00	1089.00	0.00
DIAMETER INDEX = 11	843.00	931.00	957.00	985.00	1015.00	1045.00	1075.00	1107.00	0.00
DIAMETER INDEX = 12	849.00	939.00	969.00	997.00	1029.00	1059.00	1093.00	1125.00	0.00
DIAMETER INDEX = 13	1026.00	1122.00	1152.00	1184.00	1216.00	1248.00	1282.00	1318.00	0.00
DIAMETER INDEX = 14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

D.2 LINK DATA

LINK DATA
========

TYPE 1 = COLLECTOR STREET
2 = LOCAL STREET

LINK (#)	NODE1 (#)	NODE2 (#)	LINK LENGTH (M)	FAMILY UNITS SINGLE (#)	FAMILY UNITS MULTI (#)	STREET LENGTH (M)	STREET WIDTH (M)	STREET TYPE	WALK WIDTH (M)	CURB & GUTTER LENGTH BARRIER (M)	CURB & GUTTER LENGTH ROLLED (M)	CURB & GUTTER LENGTH RLVD (M)	FRONTAGE (M)
1	2	1	126.49	0.	0.	135.6	26.0	1	1.5	140.2	106.7	0.0	0.0
2	2	3	120.40	14.	0.	240.8	10.0	2	1.2	6.1	262.1	0.0	257.5
3	3	4	161.54	0.	0.	0.0	0.0	2	0.0	0.0	0.0	9.1	0.0
4	4	5	121.92	0.	0.	0.0	0.0	2	0.0	0.0	0.0	0.0	0.0
5	5	6	38.10	0.	0.	0.0	0.0	2	0.0	0.0	0.0	0.0	0.0
6	8	4	56.39	0.	0.	0.0	0.0	2	0.0	0.0	0.0	0.0	0.0
7	7	2	89.92	0.	0.	82.3	20.0	1	1.2	91.4	67.1	0.0	0.0
8	7	8	246.89	32.	0.	179.8	20.0	1	1.2	6.1	472.4	0.0	464.8
9	12	8	74.68	0.	10.	74.7	20.0	1	1.2	3.0	134.1	0.0	0.0
10	7	10	60.96	0.	0.	61.0	20.0	1	1.2	50.3	48.8	0.0	0.0
11	14	9	76.20	0.	0.	64.0	20.0	2	1.2	6.1	125.0	0.0	0.0
12	9	10	246.89	26.	35.	245.4	20.0	2	1.2	103.6	373.4	0.0	323.1
13	10	11	112.78	0.	0.	0.0	0.0	2	0.0	0.0	0.0	0.0	0.0
14	11	12	167.64	30.	0.	347.5	10.0	2	1.2	6.1	353.6	9.1	324.9
15	12	17	80.77	0.	30.	80.8	20.0	1	1.2	9.1	125.0	0.0	0.0
16	16	17	167.64	32.	0.	347.5	10.0	2	1.2	6.1	353.6	9.1	324.9
17	17	18	120.40	9.	37.	112.8	20.0	1	1.5	6.1	217.9	0.0	103.6
18	17	29	80.77	0.	0.	82.3	20.0	1	1.2	9.1	125.0	0.0	0.0
19	15	10	71.63	0.	0.	71.6	20.0	1	1.2	88.4	54.9	0.0	0.0
20	14	15	160.02	34.	0.	221.0	20.0	1	1.2	24.4	410.0	0.0	472.4
21	13	14	54.86	4.	25.	56.4	20.0	1	1.2	3.0	105.2	0.0	0.0
22	20	13	48.77	2.	13.	48.8	20.0	1	1.2	3.0	82.3	0.0	24.4
23	19	20	134.11	0.	0.	121.9	26.0	1	1.5	6.1	243.8	0.0	0.0
24	21	20	45.72	0.	0.	45.7	20.0	1	1.2	6.1	61.0	0.0	0.0
25	21	40	70.10	15.	0.	68.6	20.0	1	1.2	3.0	126.5	0.0	161.5
26	40	41	38.10	3.	0.	38.1	20.0	1	1.2	3.0	59.4	0.0	0.0
27	21	22	83.82	9.	0.	74.7	20.0	2	1.2	9.1	135.6	0.0	143.3
28	22	23	73.15	0.	0.	65.5	20.0	2	1.2	6.1	126.5	0.0	92.7
29	23	24	123.44	27.	0.	125.0	20.0	2	1.2	3.0	245.4	0.0	251.5
30	25	24	80.77	7.	0.	82.3	20.0	2	1.2	6.1	158.5	0.0	97.5
31	22	25	152.40	25.	0.	150.9	20.0	2	1.2	6.1	286.5	0.0	381.0
32	24	34	83.82	0.	0.	0.0	0.0	2	0.0	0.0	0.0	0.0	0.0
33	41	39	99.06	7.	0.	88.4	20.0	2	1.2	9.1	153.9	0.0	89.9
34	38	39	21.34	6.	0.	53.3	15.0	2	1.2	6.1	64.0	7.6	85.3
35	39	37	82.30	7.	0.	82.3	20.0	2	1.2	6.1	131.1	0.0	76.2
36	37	36	59.44	6.	0.	149.4	10.0	2	1.2	3.0	129.5	9.1	77.7
37	36	35	85.34	0.	0.	0.0	0.0	2	0.0	0.0	0.0	0.0	0.0
38	35	33	169.16	19.	0.	161.5	20.0	2	1.2	9.1	306.3	0.0	239.3
39	34	35	59.44	6.	0.	59.4	20.0	2	1.2	3.0	120.4	0.0	48.8
40	26	34	57.91	8.	0.	57.9	20.0	2	1.2	3.0	109.7	0.0	125.0
41	27	26	82.30	0.	0.	80.8	20.0	2	1.2	91.4	59.4	0.0	36.6
42	27	29	179.83	27.	0.	173.7	20.0	2	1.2	15.2	164.6	0.0	329.2
43	29	31	82.30	0.	26.	73.2	20.0	1	1.2	9.1	128.0	0.0	0.0

44	31	30	89.92	0.	125.	100.6	26.0	1	1.5	3.0	169.2	0.0	0.0
45	32	31	83.82	7.	23.	73.2	20.0	1	1.2	3.0	155.5	0.0	89.6
46	33	32	80.77	2.	0.	82.3	20.0	1	1.2	3.0	149.4	0.0	41.5
47	28	27	57.91	14.	0.	125.0	10.0	2	1.2	6.1	134.3	9.1	121.9
48	57	33	54.86	0.	0.	54.9	20.0	1	1.2	6.1	82.3	0.0	0.0
49	55	54	76.20	0.	0.	61.0	20.0	2	1.2	12.2	109.7	0.0	216.4
50	54	61	196.60	33.	0.	196.6	20.0	2	1.2	3.0	376.4	0.0	416.0
51	58	59	42.67	0.	0.	0.0	0.0	2	0.0	0.0	0.0	0.0	0.0
52	56	57	39.62	1.	0.	38.1	20.0	1	1.2	3.0	67.1	0.0	0.0
53	51	56	277.37	26.	0.	278.9	20.0	1	1.2	187.5	356.6	0.0	399.3
54	46	51	135.64	19.	0.	135.6	20.0	1	1.2	6.1	253.0	0.0	256.0
55	45	46	64.01	13.	0.	128.0	10.0	2	1.2	6.1	163.1	9.1	201.2
56	46	44	100.58	0.	20.	100.6	20.0	1	1.2	6.1	179.8	0.0	0.0
57	44	47	164.59	23.	0.	153.9	20.0	2	1.2	9.1	297.2	0.0	277.4
58	47	37	85.34	16.	0.	77.7	20.0	2	1.2	9.1	149.4	0.0	187.4
59	43	44	57.91	0.	20.	57.9	20.0	1	1.2	6.1	56.4	0.0	0.0
60	42	43	170.69	0.	0.	163.1	20.0	1	1.5	6.1	321.6	0.0	0.0
61	41	43	94.49	6.	24.	94.5	20.0	1	1.2	6.1	163.1	0.0	89.9
62	51	50	91.44	0.	0.	82.3	26.0	1	1.2	12.2	128.0	0.0	0.0
63	48	50	67.06	13.	0.	128.0	10.0	2	1.2	6.1	157.0	9.1	198.1
64	50	49	121.92	11.	0.	120.4	26.0	1	1.5	6.1	214.9	0.0	157.0
65	50	55	175.26	29.	0.	163.1	20.0	2	1.2	9.1	306.3	0.0	153.9
66	52	53	33.53	11.	0.	88.4	10.0	2	1.2	3.0	112.8	9.1	128.0
67	53	54	86.87	12.	0.	88.4	20.0	2	1.2	6.1	160.0	0.0	121.9
68	55	60	176.78	31.	0.	178.3	20.0	2	1.2	6.1	341.4	0.0	513.6
69	60	58	103.63	13.	0.	103.6	20.0	2	1.2	6.1	199.6	0.0	115.8
70	57	58	114.30	18.	0.	106.7	20.0	2	1.2	9.1	211.8	0.0	216.4

D.3 NODE DATA

NODE DATA
========

TYPE 1 = CUL DE SAC, DEAD-END, TERMINATION
2 = DIRECTIONAL
3 = 3-WAY STREET INTERSECTION
4 = 4-WAY STREET INTERSECTION
5 = OTHER

NODE (#)	TYPE	ELEV (M)	RADIUS (M)
1	3	17.93	0.00
2	3	18.24	0.00
3	1	16.41	15.85
4	5	13.51	0.00
5	5	10.92	0.00
6	5	10.92	0.00
7	3	20.22	0.00
8	2	16.87	0.00
9	2	19.46	0.00
10	3	19.46	0.00
11	1	17.32	13.41
12	3	17.32	0.00
13	2	21.28	0.00
14	3	20.98	0.00
15	2	19.76	0.00
16	1	19.46	14.63
17	4	17.17	0.00
18	3	13.36	0.00
19	3	26.16	0.00
20	3	24.49	0.00
21	3	24.79	0.00
22	3	21.59	0.00
23	2	21.13	20.73
24	2	18.69	0.00
25	2	20.83	0.00
26	2	19.46	0.00
27	3	19.76	0.00
28	1	18.85	15.85
29	3	16.41	0.00
30	3	9.55	0.00
31	3	13.97	0.00
32	2	16.10	0.00
33	3	14.43	0.00
34	2	17.02	0.00
35	2	15.65	0.00
36	1	17.17	14.63
37	3	17.93	0.00
38	1	18.69	14.63
39	3	18.39	0.00
40	2	24.79	0.00
41	3	21.89	0.00

42	3	25.55	0.00
43	3	20.98	0.00
44	3	19.00	0.00
45	1	24.79	15.85
46	3	23.27	0.00
47	2	21.44	0.00
48	1	22.66	15.85
49	3	20.22	0.00
50	4	20.98	0.00
51	3	23.11	0.00
52	1	18.24	15.85
53	2	18.85	0.00
54	3	16.26	0.00
55	3	17.17	0.00
56	2	15.49	0.00
57	3	16.41	0.00
58	2	13.36	0.00
59	5	13.66	0.00
60	2	15.80	0.00
61	3	14.43	0.00

D.4 CALIBRATION DATA

CALIBRATION DATA VALUES

Estimated no. of people living in each unit	---	defaulted to	3.350
Estimated no. of people per acre	---	defaulted to	17.200
The min. DOMESTIC SEWER velocity within pipe	---	defaulted to	0.610
The min. STORM SEWER velocity within pipe	---	defaulted to	0.610
The max. DOMESTIC SEWER velocity within pipe	---	defaulted to	2.750
The max. STORM SEWER velocity within pipe	---	defaulted to	3.650
The min. WATER velocity within pipe	---	defaulted to	1.750
MANNING's roughness coefficient of the pipe	---	defaulted to	0.013
Min. depth of cover for DOMESTIC SEWER	---	defaulted to	2.750
Min. depth of cover for STORM SEWER	---	defaulted to	3.350
Max. spacing between manholes	---	defaulted to	100.000
Max. spacing between lights	---	defaulted to	70.000
Expected domestic water use for WATERMAIN	---	defaulted to	0.568
Safety factor for water consumption requirements	---	defaulted to	2.000

D.5 WATERMAIN TRIBUTARY DATA

--- WATERMAIN TRIBUTARY DATA ---

Feed Links				Receptor link
12	19	0	-	10
8	10	0	-	7
2	7	0	-	1
1	0	0	-	150
40	0	0	-	41
41	47	0	-	42
42	0	0	-	18
9	14	0	-	15
15	16	18	-	17
17	0	0	-	150
30	0	0	-	31
29	0	0	-	28
28	31	0	-	27
11	20	0	-	21
25	27	0	-	24
21	0	0	-	22
22	24	0	-	23
23	0	0	-	150
70	0	0	-	48
39	0	0	-	38
38	48	0	-	46
46	0	0	-	45
43	45	0	-	44
44	0	0	-	150
69	0	0	-	68
66	0	0	-	67
68	0	0	-	49
49	67	0	-	50
50	0	0	-	150
36	0	0	-	58
34	35	0	-	33
58	0	0	-	57
55	0	0	-	56
26	33	0	-	61
56	57	0	-	59
59	61	0	-	60
60	0	0	-	150
52	0	0	-	53
53	54	0	-	62
62	63	65	-	64
64	0	0	-	150

D.6 DOMESTIC SEWER TRIBUTARY DATA

--- DOMESTIC SEWER TRIBUTARY DATA ---

Feed Links				Receptor link
8	9	0	-	6
2	0	0	-	3
3	6	0	-	4
4	0	0	-	5
5	0	0	-	150
12	0	0	-	13
13	0	0	-	14
14	0	0	-	15
15	16	0	-	17
17	0	0	-	150
47	0	0	-	42
42	0	0	-	43
43	45	0	-	44
44	0	0	-	150
66	0	0	-	67
67	0	0	-	50
50	0	0	-	150
25	0	0	-	27
61	0	0	-	59
27	0	0	-	28
56	59	0	-	57
26	0	0	-	33
31	0	0	-	30
28	0	0	-	29
57	0	0	-	58
33	34	0	-	35
29	30	0	-	32
35	58	0	-	36
55	0	0	-	54
32	40	0	-	39
36	0	0	-	37
54	0	0	-	53
37	39	0	-	38
53	0	0	-	52
38	46	0	-	48
65	0	0	-	68
48	52	0	-	70
68	0	0	-	69
69	70	0	-	51
51	0	0	-	150
63	0	0	-	64
64	0	0	-	150

D.7 STORM SEWER

- STORM SEWER TRIBUTARY DATA -

Feed Links				Receptor link
7	0	0	-	1
1	0	0	-	150
8	9	0	-	6
2	0	0	-	3
3	6	0	-	4
4	0	0	-	5
5	0	0	-	150
23	24	0	-	22
22	0	0	-	21
21	0	0	-	20
20	0	0	-	19
11	0	0	-	12
10	12	19	-	13
13	0	0	-	14
14	0	0	-	15
15	16	0	-	17
17	0	0	-	150
41	47	0	-	42
18	42	0	-	43
43	45	0	-	44
44	0	0	-	150
66	0	0	-	67
65	0	0	-	49
49	67	0	-	50
50	0	0	-	150
25	0	0	-	27
60	61	0	-	59
27	0	0	-	28
56	59	0	-	57
26	0	0	-	33
31	0	0	-	30
28	0	0	-	29
57	0	0	-	58
33	34	0	-	35
29	30	0	-	32
35	58	0	-	36
32	40	0	-	39
36	0	0	-	37
37	39	0	-	38
53	0	0	-	52
38	46	0	-	48
48	52	0	-	70
68	0	0	-	69
69	70	0	-	51
51	0	0	-	150
55	0	0	-	54
54	0	0	-	62
62	63	0	-	64
64	0	0	-	150
8	9	0	-	6
2	0	0	-	3
3	6	0	-	4
4	0	0	-	5

5	0	0	-	150
12	0	0	-	13
13	0	0	-	14
14	0	0	-	15
15	16	0	-	17
17	0	0	-	150
47	0	0	-	42
42	0	0	-	43
43	45	0	-	44
44	0	0	-	150
66	0	0	-	67
67	0	0	-	50
50	0	0	-	150
25	0	0	-	27
61	0	0	-	59
27	0	0	-	28
56	59	0	-	57
26	0	0	-	33
31	0	0	-	30
28	0	0	-	29
57	0	0	-	58
33	34	0	-	35
29	30	0	-	32
35	58	0	-	36
55	0	0	-	54
32	40	0	-	39
36	0	0	-	37
54	0	0	-	53
37	39	0	-	38
53	0	0	-	52
38	46	0	-	48
65	0	0	-	68
48	52	0	-	70
68	0	0	-	69
69	70	0	-	51
51	0	0	-	150
63	0	0	-	64
64	0	0	-	150

D.8 WATERMAIN LINK DATA

WATERMAIN
========

LINK (#)	TRIBUTARY POPULATION (#.	FLOW RATE (CMS)	REQUIRED DIAMETER (M)
12	204.	0.03164	0.15173
19	1.	0.00205	0.03859
8	107.	0.02241	0.12768
10	205.	0.03173	0.15193
2	47.	0.01452	0.10279
7	313.	0.03988	0.17034
1	359.	0.04307	0.17702
40	27.	0.01087	0.08893
41	27.	0.01087	0.08893
47	47.	0.01452	0.10279
42	164.	0.02812	0.14304
9	34.	0.01220	0.09420
14	101.	0.02165	0.12552
15	235.	0.03409	0.15749
16	107.	0.02241	0.12768
18	165.	0.02821	0.14327
17	661.	0.06057	0.20993
30	23.	0.01015	0.08592
29	90.	0.02048	0.12207
28	90.	0.02048	0.12707
31	107.	0.02241	0.12768
11	1.	0.00205	0.03859
20	114.	0.02314	0.12975
25	50.	0.01505	0.10465
27	228.	0.03356	0.15626
21	212.	0.03228	0.15325
22	262.	0.03623	0.16236
24	278.	0.03741	0.16497
23	540.	0.05405	0.19831
70	60.	0.01655	0.10975
39	20.	0.00937	0.08258
38	84.	0.01967	0.11962
48	60.	0.01655	0.10975
46	151.	0.02687	0.13981
43	87.	0.02008	0.12087
45	251.	0.03539	0.16047
44	757.	0.06545	0.21822
69	44.	0.01397	0.10083
66	37.	0.01281	0.09655
68	147.	0.02655	0.13898
49	147.	0.02655	0.13898
67	77.	0.01882	0.11703
50	335.	0.04143	0.17362
36	20.	0.00937	0.08258
34	20.	0.00937	0.08258
35	23.	0.01015	0.08592
58	74.	0.01839	0.11567
55	44.	0.01397	0.10083

26	10.	0.00658	0.06917
33	67.	0.01749	0.11281
56	111.	0.02278	0.12873
57	151.	0.02687	0.13981
59	328.	0.04097	0.17265
61	178.	0.02933	0.14609
60	506.	0.05209	0.19467
52	3.	0.00377	0.05235
53	90.	0.02048	0.12207
54	64.	0.01703	0.11131
62	154.	0.02719	0.14064
63	44.	0.01397	0.10083
65	97.	0.02127	0.12440
64	332.	0.04120	0.17314

D.9 DOMESTIC SEWER LINK DATA

DOMESTIC SEWER
==========

MINIMUM ACCEPTABLE PIPE VELOCITY (M/S) = 0.610
MAXIMUM ACCEPTABLE PIPE VELOCITY (M/S) = 2.750

MINIMUM DEPTH OF GROUND COVER OVER PIPE (M) = 2.750

LINK (#)	TRIBUTARY POPULATION (#)	FLOW RATE (CMS)	GROUND SLOPE (M/M)	CALCULATED DIAMETER (M)	VELOCITY (M/S)	PIPE SLOPE (M/M)	DEPTH OF EXCAVATION NODE1 (M)	DEPTH OF EXCAVATION NODE2 (M)	AVERAGE DEPTH (M)	LINK (#)
8	107.	0.00282	0.01357	0.07529	0.71067	0.01357	3.05000	3.05000	3.05000	8
9	34.	0.00088	0.00603	0.05667	0.61061	0.02370	3.05000	4.36965	3.70982	9
2	47.	0.00123	0.01520	0.05406	0.61061	0.01768	3.05000	3.34888	3.19944	2
3	47.	0.00123	0.01795	0.05239	0.61363	0.01795	3.34888	3.34888	3.34888	3
6	141.	0.00370	0.05959	0.06317	1.29792	0.05959	4.36965	4.36965	4.36965	6
4	188.	0.00494	0.02124	0.08538	0.98159	0.02124	4.36965	4.36965	4.36965	4
5	188.	0.00494	0.00000	0.21134	0.61061	0.00595	4.36965	4.59625	4.48295	5
12	204.	0.00538	0.00000	0.21822	0.61061	0.00555	3.10000	4.46957	3.78479	12
13	204.	0.00538	0.01897	0.09005	0.96852	0.01897	4.46957	4.46957	4.46957	13
14	305.	0.00803	0.00000	0.25354	0.61061	0.00423	4.46957	5.17840	4.82399	14
15	405.	0.01067	0.00186	0.18000	0.61061	0.00320	5.17840	5.28674	5.23257	15
16	107.	0.00282	0.01366	0.07519	0.71306	0.01366	3.05000	3.05000	3.05000	16
17	667.	0.01755	0.03164	0.12747	1.62649	0.03164	5.28674	5.28674	5.28674.	17
47	47.	0.00123	0.01571	0.05372	0.61061	0.01768	3.05000	3.16395	3.10697	47
42	137.	0.00362	0.01863	0.07785	0.85637	0.01863	3.05000	3.05000	3.05000	42
43	224.	0.00591	0.02965	0.08579	1.16573	0.02965	3.05000	3.05000	3.05000	43
45	101.	0.00265	0.02541	0.06533	0.87171	0.02541	3.05000	3.05000	3.05000	45
44	744.	0.01958	0.04915	0.12228	1.96420	0.04915	3.05000	3.05000	3.05000	44
66	37.	0.00097	0.01819	0.04774	0.61061	0.02177	3.05000	3.16985	3.10992	66
67	77.	0.00203	0.02981	0.05739	0.85022	0.02981	3.05000	3.05000	3.05000	67
50	188.	0.00494	0.00931	0.09966	0.73354	0.00931	3.05000	3.05000	3.05000	50
25	50.	0.00132	0.00000	0.12896	0.61061	0.01670	3.05000	4.22099	3.63549	25
61	101.	0.00265	0.00963	0.07836	0.61947	0.00963	3.05000	3.05000	3.05000	61
27	80.	0.00212	0.03818	0.05567	0.93930	0.03818	4.22099	4.22099	4.22099	27
56	67.	0.00176	0.04245	0.05097	0.92341	0.04245	3.05000	3.05000	3.05000	56
59	168.	0.00441	0.03419	0.07484	1.12450	0.03419	3.05000	3.05000	3.05000	59
26	10.	0.00026	0.07612	0.02243	0.63628	0.07612	3.05000	3.05000	3.05000	26
31	84.	0.00221	0.00499	0.08280	0.61061	0.01077	3.05000	3.93180	3.49090	31
28	80.	0.00212	0.00629	0.07807	0.61061	0.01115	4.22099	4.57691	4.39895	28
57	312.	0.00820	0.01482	0.11047	1.00020	0.01482	3.05000	3.05000	3.05000	57
33	34.	0.00088	0.03533	0.04068	0.70188	0.03533	3.05000	3.05000	3.05000	33
34	20.	0.00053	0.01406	0.03992	0.61061	0.03693	3.05000	3.53816	3.29408	34
29	171.	0.00450	0.01977	0.08356	0.93171	0.01977	4.57691	4.57691	4.57691	29
30	107.	0.00282	0.02649	0.06641	0.90048	0.02649	3.93180	3.93180	3.93180	30
35	77.	0.00203	0.00559	0.07855	0.61061	0.01156	3.53816	4.02970	3.78393	35
58	365.	0.00961	0.04113	0.09683	1.50655	0.04113	3.05000	3.05000	3.05000	58
55	44.	0.00115	0.02375	0.04835	0.66161	0.02375	3.05000	3.05000	3.05000	55
32	278.	0.00732	0.01992	0.10015	1.07760	0.01992	4.57691	4.57691	4.57691	32
40	27.	0.00071	0.04213	0.03620	0.69867	0.04213	3.05000	3.05000	3.05000	40
36	462.	0.01217	0.01279	0.13170	1.05779	0.01279	4.02970	4.02970	4.02970	36
54	107.	0.00282	0.00118	0.11902	0.61061	0.00878	3.05000	4.08056	3.56528	54
37	462.	0.01217	0.01781	0.12376	1.19411	0.01781	4.02970	4.02970	4.02970	37

39	325.	0.00856	0.02305	0.10332	1.18471	0.02305	4.57691	4.57691	4.57691	39
53	194.	0.00512	0.02747	0.08244	1.08634	0.02747	4.08056	4.08056	4.08056	53
38	851.	0.02241	0.00721	0.18431	0.98613	0.00721	4.57691	4.57691	4.57691	38
46	7.	0.00018	0.02068	0.02460	0.61061	0.09684	3.05000	9.20149	6.12574	46
65	97.	0.00256	0.02174	0.06642	0.81567	0.02174	3.05000	3.05000	3.05000	65
48	858.	0.02258	0.03609	0.13668	1.82398	0.03609	3.05000	3.05000	3.05000	48
52	198.	0.00520	0.02322	0.08563	1.02896	0.02322	3.05000	3.05000	3.05000	52
68	201.	0.00529	0.00775	0.10585	0.70017	0.00775	3.05000	3.05000	3.05000	68
69	245.	0.00644	0.02355	0.09250	1.09961	0.02355	3.05000	3.05000	3.05000	69
70	1116.	0.02874	0.02668	0.15833	1.73553	0.02668	3.05000	3.05000	3.05000	70
51	1360.	0.03368	0.00703	0.21577	1.09032	0.00703	3.10000	3.10000	3.10000	51
63	44.	0.00115	0.02505	0.04787	0.67349	0.02505	3.05000	3.05000	3.05000	63
64	80.	0.00212	0.00623	0.07820	0.61061	0.01115	3.05000	3.64990	3.34995	64

D.10 STORM SEWER LINK DATA

STORM SEWER
==========

MINIMUM ACCEPTABLE PIPE VELOCITY (M/S) = 0.610
MAXIMUM ACCEPTABLE PIPE VELOCITY (M/S) = 3.650

MINIMUM DEPTH OF GROUND COVER OVER PIPE (M) = 3.350

LINK (#)	TRIBUTARY POPULATION (HA)	FLOW RATE (CMS)	GROUND SLOPE (M/M)	CALCULATED DIAMETER (M)	VELOCITY (M/S)	PIPE SLOPE (M/M)	DEPTH OF EXCAVATION NODE1 (M)	DEPTH OF EXCAVATION NODE2 (M)	AVERAGE DEPTH (M)	LINK (#)
7	0.32000	0.01472	0.02202	0.12772	1.30953	0.02202	3.75000	3.75000	3.75000	7
1	0.88000	0.03144	0.00245	0.25622	0.72208	0.00245	3.75000	3.75000	3.75000	1
8	1.26000	0.04115	0.01357	0.20564	1.46880	0.01357	3.75000	3.75000	3.75000	8
9	0.90000	0.03197	0.00603	0.21782	1.01872	0.00603	3.75000	3.75000	3.75000	9
2	0.82000	0.02982	0.01520	0.17840	1.40206	0.01520	3.75000	3.75000	3.75000	2
3	2.10000	0.06036	0.01795	0.22527	1.80170	0.01795	3.75000	3.75000	3.75000	3
6	2.29000	0.06441	0.05959	0.18433	2.84591	0.05959	3.75000	3.75000	3.75000	6
4	5.05000	0.11656	0.02124	0.27936	2.22948	0.02124	3.75000	3.75000	3.75000	4
5	5.21000	0.11932	0.00000	0.69760	0.61061	0.00064	4.20000	4.22431	4.21215	5
23	0.63000	0.02447	0.01245	0.17196	1.23581	0.01245	3.75000	3.75000	3.75000	23
24	0.18000	0.00956	0.00656	0.13633	0.75173	0.00656	3.75000	3.75000	3.75000	24
22	1.09000	0.03691	0.06582	0.14683	2.51885	0.06582	3.75000	3.75000	3.75000	22
21	1.53000	0.04760	0.00547	0.25752	1.08349	0.00547	3.75000	3.75000	3.75000	21
20	2.66000	0.07207	0.00762	0.28269	1.34187	0.00762	3.75000	3.75000	3.75000	20
11	1.07000	0.03640	0.01995	0.18271	1.63491	0.01995	3.75000	3.75000	3.75000	11
10	0.19000	0.00996	0.01247	0.12272	0.95359	0.01247	3.75000	3.75000	3.75000	10
12	3.20000	0.08278	0.00000	0.60823	0.61061	0.00084	4.20000	4.40665	4.30332	12
19	3.46000	0.08778	0.00419	0.34057	1.13418	0.00419	3.82500	3.82500	3.82500	19
13	7.16000	0.15145	0.01897	0.31478	2.30946	0.01897	4.40665	4.40665	4.40665	13
14	8.11000	0.16628	0.00000	0.79006	0.61061	0.00051	4.40665	4.49257	4.44961	14
15	8.94000	0.17889	0.00186	0.51806	0.97518	0.00186	4.49257	4.49257	4.49257	15
16	0.91000	0.03224	0.01366	0.18742	1.37793	0.01366	3.75000	3.75000	3.75000	16
17	11.03000	0.20942	0.03164	0.32296	3.03108	0.03164	4.49257	4.49257	4.49257	17
41	0.35000	0.01574	0.00365	0.18351	0.70191	0.00365	3.75000	3.75000	3.75000	41
47	0.50000	0.02057	0.01571	0.15426	1.27933	0.01571	3.75000	3.75000	3.75000	47
18	0.89000	0.03171	0.00941	0.19973	1.19778	0.00941	3.75000	3.75000	3.75000	18
42	1.84000	0.05466	0.01863	0.21555	1.78057	0.01863	3.75000	3.75000	3.75000	42
43	3.41000	0.08683	0.02965	0.23500	2.37857	0.02965	3.75000	3.75000	3.75000	43
45	0.73000	0.02733	0.02541	0.15680	1.64581	0.02541	3.75000	3.75000	3.75000	45
44	5.91000	0.13115	0.04915	0.24950	3.18732	0.04915	3.75000	3.75000	3.75000	44
66	0.36000	0.01608	0.01819	0.13684	1.25469	0.01819	3.75000	3.75000	3.75000	66
65	0.85000	0.03063	0.02174	0.16852	1.60617	0.02174	3.75000	3.75000	3.75000	65
49	1.42000	0.04501	0.01194	0.21782	1.43415	0.01194	3.75000	3.75000	3.75000	49
67	0.82000	0.02982	0.02981	0.15723	1.78610	0.02981	3.75000	3.75000	3.75000	67
50	3.29000	0.08452	0.00931	0.28908	1.49496	0.00931	3.75000	3.75000	3.75000	50
25	0.41000	0.01773	0.00000	0.34126	0.61061	0.00238	3.82500	3.99194	3.90847	25
60	0.97000	0.03382	0.02677	0.16819	1.78248	0.02677	3.75000	3.75000	3.75000	60
61	0.59000	0.02329	0.00963	0.17715	1.11097	0.00963	3.75000	3.75000	3.75000	61
27	0.80000	0.02927	0.03818	0.14907	1.93861	0.03818	3.99194	3.99194	3.99194	27
56	1.14000	0.03817	0.04245	0.16143	2.17518	0.04245	3.75000	3.75000	3.75000	56
59	2.03000	0.05884	0.03419	0.19774	2.26665	0.03419	3.75000	3.75000	3.75000	59
26	0.46000	0.01933	0.07612	0.11210	2.19854	0.07612	3.75000	3.75000	3.75000	26

31	0.82000	0.02982	0.00499	0.21986	0.93217	0.00499	3.75000	3.75000	3.75000	31
28	1.43000	0.04525	0.00629	0.24614	1.12964	0.00629	3.99194	3.99194	3.99194	28
57	3.97000	0.09731	0.01482	0.27930	1.86245	0.01482	3.75000	3.75000	3.75000	57
33	1.04000	0.03563	0.03533	0.16283	1.99866	0.03533	3.75000	3.75000	3.75000	33
34	0.22000	0.01111	0.01406	0.12504	1.02679	0.01406	3.75000	3.75000	3.75000	34
29	2.12000	0.06079	0.01977	0.22183	1.86997	0.01977	3.99194	3.99194	3.99194	29
30	1.53000	0.04760	0.02649	0.19157	1.95019	0.02649	3.75000	3.75000	3.75000	30
35	1.65000	0.05037	0.00559	0.26198	1.10476	0.00559	3.75000	3.75000	3.75000	35
58	4.80000	0.11221	0.04113	0.24332	2.86921	0.04113	3.75000	3.75000	3.75000	58
32	5.57000	0.12545	0.01992	0.29065	2.19161	0.01992	3.99194	3.99194	3.99194	32
40	0.29000	0.01367	0.04213	0.11000	1.60925	0.04213	3.75000	3.75000	3.75000	40
36	7.08000	0.15018	0.01279	0.33790	1.97194	0.01279	3.82500	3.82500	3.82500	36
37	7.30000	0.15367	0.01781	0.32028	2.25885	0.01781	3.82500	3.82500	3.82500	37
39	6.11000	0.13447	0.02305	0.29027	2.35413	0.02305	3.99194	3.99194	3.99194	39
53	1.36000	0.04358	0.02747	0.18407	1.92693	0.02747	3.75000	3.75000	3.75000	53
38	14.24000	0.25364	0.00721	0.45789	1.82242	0.00721	3.99194	3.99194	3.99194	38
46	1.15000	0.03842	0.02068	0.18520	1.68117	0.02068	3.75000	3.75000	3.75000	46
48	15.94000	0.27603	0.03609	0.34947	3.37212	0.03609	3.82500	3.82500	3.82500	48
52	1.51000	0.04713	0.02322	0.19564	1.85407	0.02322	3.82500	3.82500	3.82500	52
68	1.02000	0.03512	0.00775	0.21523	1.14614	0.00775	3.75000	3.75000	3.75000	68
69	3.76000	0.09343	0.02355	0.25221	2.21699	0.02355	3.75000	3.75000	3.75000	69
70	18.49000	0.30852	0.02668	0.38559	3.12927	0.02668	3.90000	3.90000	3.90000	70
51	22.25000	0.35447	0.00703	0.52161	1.89886	0.00703	3.97500	3.97500	3.97500	51
55	0.65000	0.02505	0.02375	0.15370	1.56651	0.02375	3.75000	3.75000	3.75000	55
54	1.89000	0.05577	0.00118	0.36436	0.61880	0.00118	3.82500	3.82500	3.82500	54
62	2.49000	0.06859	0.02329	0.22506	2.05232	0.02329	3.82500	3.82500	3.82500	62
63	0.45000	0.01901	0.02505	0.13722	1.47585	0.02505	3.75000	3.75000	3.75000	63
64	3.60000	0.09043	0.00623	0.31965	1.33632	0.00623	3.82500	3.82500	3.82500	64

D.11 STORM SEWER SYSTEM DATA

T O R M S E W E R S Y S T E M

LINK	PIPE LENGTH	DIAMETER	UNIT COST	TOTAL COST
1	126.49m	300.00mm	$ 468.00	$ 59197.32
2	120.40m	300.00mm	$ 468.00	$ 56347.20
3	161.54m	300.00mm	$ 468.00	$ 75600.72
4	121.92m	300.00mm	$ 468.00	$ 57058.56
5	38.10m	750.00mm	$ 700.00	$ 26670.00
6	56.39m	300.00mm	$ 468.00	$ 26390.52
7	89.92m	300.00mm	$ 468.00	$ 42082.56
8	246.89m	300.00mm	$ 468.00	$ 115544.52
9	74.68m	300.00mm	$ 468.00	$ 34950.24
10	60.96m	300.00mm	$ 468.00	$ 28529.28
11	76.20m	300.00mm	$ 468.00	$ 35661.60
12	246.89m	750.00mm	$ 700.00	$ 172823.00
13	112.78m	375.00mm	$ 490.00	$ 55262.20
14	167.64m	900.00mm	$ 876.00	$ 146852.64
15	80.77m	525.00mm	$ 682.00	$ 55085.14
16	167.64m	300.00mm	$ 468.00	$ 78455.52
17	120.40m	375.00mm	$ 490.00	$ 58996.00
18	80.77m	300.00mm	$ 468.00	$ 37800.36
19	71.63m	375.00mm	$ 474.00	$ 33952.62
20	160.02m	300.00mm	$ 468.00	$ 74889.36
21	54.86m	300.00mm	$ 468.00	$ 25674.48
22	48.77m	300.00mm	$ 468.00	$ 22824.36
23	134.11m	300.00mm	$ 468.00	$ 62763.48
24	45.72m	300.00mm	$ 468.00	$ 21396.96
25	70.10m	375.00mm	$ 474.00	$ 33227.40
26	38.10m	300.00mm	$ 468.00	$ 17830.80
27	83.82m	300.00mm	$ 468.00	$ 39227.76
28	73.15m	300.00mm	$ 468.00	$ 34234.20
29	123.44m	300.00mm	$ 468.00	$ 57769.92
30	80.77m	300.00mm	$ 468.00	$ 37800.36
31	152.40m	300.00mm	$ 468.00	$ 71323.20
32	83.82m	300.00mm	$ 468.00	$ 39227.76
33	99.06m	300.00mm	$ 468.00	$ 46360.08
34	21.34m	300.00mm	$ 468.00	$ 9987.12
35	82.30m	300.00mm	$ 468.00	$ 38516.40
36	59.44m	375.00mm	$ 474.00	$ 28174.56
37	85.34m	375.00mm	$ 474.00	$ 40451.16
38	169.16m	525.00mm	$ 664.00	$ 112322.24
39	59.44m	300.00mm	$ 468.00	$ 27817.92
40	57.91m	300.00mm	$ 468.00	$ 27101.88
41	82.30m	300.00mm	$ 468.00	$ 38516.40
42	179.83m	300.00mm	$ 468.00	$ 84160.44
43	82.30m	300.00mm	$ 468.00	$ 38516.40

44	89.92m	300.00mm	$ 468.00	$ 42082.56	
45	83.82m	300.00mm	$ 468.00	$ 39227.76	
46	80.77m	300.00mm	$ 468.00	$ 37800.36	
47	57.91m	300.00mm	$ 468.00	$ 27101.88	
48	54.86m	375.00mm	$ 474.00	$ 26003.64	
49	76.20m	300.00mm	$ 468.00	$ 35661.60	
50	196.60m	300.00mm	$ 468.00	$ 92008.80	
51	42.67m	525.00mm	$ 664.00	$ 28332.88	
52	39.62m	300.00mm	$ 468.00	$ 18542.16	
53	277.37m	300.00mm	$ 468.00	$ 129809.16	
54	135.64m	375.00mm	$ 474.00	$ 64293.36	
55	64.01m	300.00mm	$ 468.00	$ 29956.68	
56	100.58m	300.00mm	$ 468.00	$ 47071.44	
57	164.59m	300.00mm	$ 468.00	$ 77028.12	
58	85.34m	300.00mm	$ 468.00	$ 39939.12	
59	57.91m	300.00mm	$ 468.00	$ 27101.88	
60	170.69m	300.00mm	$ 468.00	$ 79882.92	
61	94.49m	300.00mm	$ 468.00	$ 44221.32	
62	91.44m	300.00mm	$ 468.00	$ 42793.92	
63	67.06m	300.00mm	$ 468.00	$ 31384.08	
64	121.92m	375.00mm	$ 474.00	$ 57790.08	
65	175.26m	300.00mm	$ 468.00	$ 82021.68	
66	33.53m	300.00mm	$ 468.00	$ 15692.04	
67	86.87m	300.00mm	$ 468.00	$ 40655.16	
68	176.78m	300.00mm	$ 468.00	$ 82733.04	
69	103.63m	300.00mm	$ 468.00	$ 48498.84	
70	114.30m	450.00mm	$ 480.00	$ 54864.00	
TOTAL	7193.29m			$ 3569873.00	

D.12 DOMESTIC SEWER SYSTEM DATA

D O M E S T I C S E W E R S Y S T E M

LINK	PIPE LENGTH	DIAMETER	UNIT COST	TOTAL COST
1	0.00m	0.00mm	$ 0.00	$ 0.00
2	120.40m	200.00mm	$ 310.00	$ 37324.00
3	161.54m	200.00mm	$ 310.00	$ 50077.40
4	121.92m	200.00mm	$ 340.00	$ 41452.80
5	38.10m	250.00mm	$ 344.00	$ 13106.40
6	56.39m	200.00mm	$ 340.00	$ 19172.60
7	0.00m	0.00mm	$ 0.00	$ 0.00
8	246.89m	200.00mm	$ 310.00	$ 76535.90
9	74.68m	200.00mm	$ 324.00	$ 24196.32
10	0.00m	0.00mm	$ 0.00	$ 0.00
11	0.00m	0.00mm	$ 0.00	$ 0.00
12	246.89m	250.00mm	$ 328.00	$ 80979.92
13	112.78m	200.00mm	$ 340.00	$ 38345.20
14	167.64m	300.00mm	$ 406.00	$ 68061.84
15	80.77m	200.00mm	$ 372.00	$ 30046.44
16	167.64m	200.00mm	$ 310.00	$ 51968.40
17	120.40m	200.00mm	$ 372.00	$ 44788.80
18	0.00m	0.00mm	$ 0.00	$ 0.00
19	0.00m	0.00mm	$ 0.00	$ 0.00
20	0.00m	0.00mm	$ 0.00	$ 0.00
21	0.00m	0.00mm	$ 0.00	$ 0.00
22	0.00m	0.00mm	$ 0.00	$ 0.00
23	0.00m	0.00mm	$ 0.00	$ 0.00
24	0.00m	0.00mm	$ 0.00	$ 0.00
25	70.10m	200.00mm	$ 324.00	$ 22712.40
26	38.10m	200.00mm	$ 310.00	$ 11811.00
27	83.82m	200.00mm	$ 340.00	$ 28498.80
28	73.15m	200.00mm	$ 340.00	$ 24871.00
29	123.44m	200.00mm	$ 356.00	$ 43944.64
30	80.77m	200.00mm	$ 324.00	$ 26169.48
31	152.40m	200.00mm	$ 310.00	$ 47244.00
32	83.82m	200.00mm	$ 356.00	$ 29839.92
33	99.06m	200.00mm	$ 310.00	$ 30708.60
34	21.34m	200.00mm	$ 310.00	$ 6615.40
35	82.30m	200.00mm	$ 324.00	$ 26665.20
36	59.44m	200.00mm	$ 340.00	$ 20209.60
37	85.34m	200.00mm	$ 340.00	$ 29015.60
38	169.16m	200.00mm	$ 356.00	$ 60220.96
39	59.44m	200.00mm	$ 356.00	$ 21160.64
40	57.91m	200.00mm	$ 310.00	$ 17952.10
41	0.00m	0.00mm	$ 0.00	$ 0.00
42	179.83m	200.00mm	$ 310.00	$ 55747.30
43	82.30m	200.00mm	$ 310.00	$ 25513.00

44	89.92m	200.00mm	$ 310.00	$ 27875.20
45	83.82m	200.00mm	$ 310.00	$ 25984.20
46	80.77m	200.00mm	$ 410.00	$ 33115.70
47	57.91m	200.00mm	$ 310.00	$ 17952.10
48	54.86m	200.00mm	$ 310.00	$ 17006.60
49	0.00m	0.00mm	$ 0.00	$ 0.00
50	196.60m	200.00mm	$ 310.00	$ 60946.00
51	42.67m	250.00mm	$ 314.00	$ 13398.38
52	39.62m	200.00mm	$ 310.00	$ 12282.20
53	277.37m	200.00mm	$ 340.00	$ 94305.80
54	135.64m	200.00mm	$ 324.00	$ 43947.36
55	64.01m	200.00mm	$ 310.00	$ 19843.10
56	100.58m	200.00mm	$ 310.00	$ 31179.80
57	164.59m	200.00mm	$ 310.00	$ 51022.90
58	85.34m	200.00mm	$ 310.00	$ 26455.40
59	57.91m	200.00mm	$ 310.00	$ 17952.10
60	0.00m	0.00mm	$ 0.00	$ 0.00
61	94.49m	200.00mm	$ 310.00	$ 29291.90
62	0.00m	0.00mm	$ 0.00	$ 0.00
63	67.06m	200.00mm	$ 310.00	$ 20788.60
64	121.92m	200.00mm	$ 310.00	$ 37795.20
65	175.26m	200.00mm	$ 310.00	$ 54330.60
66	33.53m	200.00mm	$ 310.00	$ 10394.30
67	86.87m	200.00mm	$ 310.00	$ 26929.70
68	176.78m	200.00mm	$ 310.00	$ 54801.80
69	103.63m	200.00mm	$ 310.00	$ 32125.30
70	114.30m	200.00mm	$ 310.00	$ 35433.00
TOTAL	5823.21m			$ 1900113.13

D.13 WATERMAIN SYSTEM

WATERMAIN SYSTEM

LINK	PIPE LENGTH	DIAMETER	UNIT COST	TOTAL COST
1	126.49m	200.00mm	$ 165.00	$ 20870.85
2	120.40m	150.00mm	$ 140.00	$ 16856.00
3	0.00m	0.00mm	$ 0.00	$ 0.00
4	0.00m	0.00mm	$ 0.00	$ 0.00
5	0.00m	0.00mm	$ 0.00	$ 0.00
6	0.00m	0.00mm	$ 0.00	$ 0.00
7	89.92m	200.00mm	$ 165.00	$ 14836.80
8	246.89m	150.00mm	$ 140.00	$ 34564.60
9	74.68m	150.00mm	$ 140.00	$ 10455.20
10	60.96m	200.00mm	$ 165.00	$ 10058.40
11	76.20m	150.00mm	$ 140.00	$ 10668.00
12	246.89m	200.00mm	$ 165.00	$ 40736.85
13	0.00m	0.00mm	$ 0.00	$ 0.00
14	167.64m	150.00mm	$ 140.00	$ 23469.60
15	80.77m	200.00mm	$ 165.00	$ 13327.05
16	167.64m	150.00mm	$ 140.00	$ 23469.60
17	120.40m	250.00mm	$ 180.00	$ 21672.00
18	80.77m	150.00mm	$ 140.00	$ 11307.80
19	71.63m	150.00mm	$ 140.00	$ 10028.20
20	160.02m	150.00mm	$ 140.00	$ 22402.80
21	54.86m	200.00mm	$ 165.00	$ 9051.90
22	48.77m	200.00mm	$ 165.00	$ 8047.05
23	134.11m	200.00mm	$ 165.00	$ 22128.15
24	45.72m	200.00mm	$ 165.00	$ 7543.80
25	70.10m	150.00mm	$ 140.00	$ 9814.00
26	38.10m	150.00mm	$ 140.00	$ 5334.00
27	83.82m	200.00mm	$ 165.00	$ 13830.30
28	73.15m	150.00mm	$ 140.00	$ 10241.00
29	123.44m	150.00mm	$ 140.00	$ 17281.60
30	80.77m	150.00mm	$ 140.00	$ 11307.80
31	152.40m	150.00mm	$ 140.00	$ 21336.00
32	0.00m	0.00mm	$ 0.00	$ 0.00
33	99.06m	150.00mm	$ 140.00	$ 13868.40
34	21.34m	150.00mm	$ 140.00	$ 2987.60
35	82.30m	150.00mm	$ 140.00	$ 11522.00
36	59.44m	150.00mm	$ 140.00	$ 8321.60
37	0.00m	0.00mm	$ 0.00	$ 0.00
38	169.16m	150.00mm	$ 140.00	$ 23682.40
39	59.44m	150.00mm	$ 140.00	$ 8321.60
40	57.91m	150.00mm	$ 140.00	$ 8107.40
41	82.30m	150.00mm	$ 140.00	$ 11522.00
42	179.83m	150.00mm	$ 140.00	$ 25176.20
43	82.30m	150.00mm	$ 140.00	$ 11522.00

44	89.92m	250.00mm	$ 180.00	$ 16185.60
45	83.82m	200.00mm	$ 165.00	$ 13830.30
46	80.77m	150.00mm	$ 140.00	$ 11307.80
47	57.91m	150.00mm	$ 140.00	$ 8107.40
48	54.86m	150.00mm	$ 140.00	$ 7680.40
49	76.20m	150.00mm	$ 140.00	$ 10668.00
50	196.60m	200.00mm	$ 165.00	$ 32439.00
51	0.00m	0.00mm	$ 0.00	$ 0.00
52	39.62m	150.00mm	$ 140.00	$ 5546.80
53	277.37m	150.00mm	$ 140.00	$ 38831.80
54	135.64m	150.00mm	$ 140.00	$ 18989.60
55	64.01m	150.00mm	$ 140.00	$ 8961.40
56	100.58m	150.00mm	$ 140.00	$ 14081.20
57	164.59m	150.00mm	$ 140.00	$ 23042.60
58	85.34m	150.00mm	$ 140.00	$ 11947.60
59	57.91m	200.00mm	$ 165.00	$ 9555.15
60	170.69m	200.00mm	$ 165.00	$ 28163.85
61	94.49m	150.00mm	$ 140.00	$ 13228.60
62	91.44m	150.00mm	$ 140.00	$ 12801.60
63	67.06m	150.00mm	$ 140.00	$ 9388.40
64	121.92m	200.00mm	$ 165.00	$ 20116.80
65	175.26m	150.00mm	$ 140.00	$ 24536.40
66	33.53m	150.00mm	$ 140.00	$ 4694.20
67	86.87m	150.00mm	$ 140.00	$ 12161.80
68	176.78m	150.00mm	$ 140.00	$ 24749.20
69	103.63m	150.00mm	$ 140.00	$ 14508.20
70	114.30m	150.00mm	$ 140.00	$ 16002.00
TOTAL	6490.73m			$ 957196.44

D.14 MANHOLES AND CATCHBASINS

M A N H O L E S & C A T C H - B A S I N S

LINK	DOMESTIC SEWER MANHOLE QUANTITY	DOMESTIC SEWER MANHOLE COST	STORM SEWER MANHOLE QUANTITY	STORM SEWER MANHOLE COST	STORM SEWER CATCH-BASIN QUANTITY	STORM SEWER CATCH-BASIN COST
1	0	$ 0.00	1	$ 3804.00	2	$ 18384.12
2	1	$ 3515.19	1	$ 3804.00	1	$ 5453.56
3	1	$ 3515.19	1	$ 3804.00	0	$ 0.00
4	0	$ 0.00	0	$ 0.00	0	$ 0.00
5	2	$ 8663.45	2	$ 8273.50	0	$ 0.00
6	1	$ 4250.15	1	$ 3804.00	0	$ 0.00
7	1	$ 3300.00	1	$ 3804.00	2	$ 15579.13
8	1	$ 3300.00	1	$ 3804.00	4	$ 31158.26
9	1	$ 4250.15	1	$ 3804.00	0	$ 0.00
10	1	$ 3300.00	1	$ 3804.00	1	$ 6079.81
11	0	$ 0.00	0	$ 0.00	0	$ 0.00
12	2	$ 7071.41	2	$ 8316.26	4	$ 33258.25
13	0	$ 0.00	0	$ 0.00	0	$ 0.00
14	2	$ 8948.62	2	$ 8590.48	3	$ 17722.50
15	1	$ 4832.45	1	$ 4338.65	1	$ 6595.89
16	2	$ 6600.00	2	$ 7608.00	5	$ 27309.62
17	1	$ 4910.46	1	$ 4338.65	2	$ 16728.21
18	1	$ 4910.46	1	$ 4338.65	1	$ 6079.81
19	1	$ 4322.09	1	$ 4276.79	2	$ 15662.83
20	0	$ 0.00	2	$ 7662.00	2	$ 15579.13
21	0	$ 0.00	1	$ 3804.00	2	$ 15579.13
22	0	$ 0.00	1	$ 3804.00	0	$ 0.00
23	0	$ 0.00	1	$ 3804.00	1	$ 7481.41
24	0	$ 0.00	1	$ 3804.00	2	$ 15579.13
25	1	$ 4143.11	1	$ 3978.20	0	$ 0.00
26	1	$ 3300.00	1	$ 3858.00	2	$ 15579.13
27	0	$ 0.00	0	$ 0.00	2	$ 15849.13
28	0	$ 0.00	0	$ 0.00	0	$ 0.00
29	1	$ 4399.37	1	$ 3978.20	2	$ 15847.67
30	1	$ 4399.37	1	$ 3978.20	0	$ 0.00
31	3	$ 11794.61	3	$ 11586.20	0	$ 0.00
32	0	$ 0.00	0	$ 0.00	0	$ 0.00
33	0	$ 0.00	0	$ 0.00	1	$ 6079.81
34	1	$ 3300.00	1	$ 3804.00	3	$ 19865.15
35	1	$ 3651.47	1	$ 3804.00	1	$ 6079.81
36	1	$ 4005.38	1	$ 3858.00	1	$ 5495.41
37	0	$ 0.00	0	$ 0.00	0	$ 0.00
38	1	$ 4399.37	1	$ 3978.20	2	$ 15849.13
39	0	$ 0.00	0	$ 0.00	2	$ 15849.13
40	1	$ 4399.37	1	$ 3978.20	2	$ 15579.13
41	2	$ 6600.00	2	$ 7608.00	0	$ 0.00

42	1	$ 3300.00	1	$ 3804.00	4	$ 31158.26
43	1	$ 3300.00	1	$ 3804.00	1	$ 6079.81
44	1	$ 3300.00	1	$ 3804.00	2	$ 18384.12
45	1	$ 3300.00	1	$ 3804.00	4	$ 31158.26
46	1	$ 3300.00	1	$ 3804.00	0	$ 0.00
47	1	$ 3382.04	1	$ 3804.00	1	$ 5453.56
48	1	$ 7729.07	1	$ 3978.20	0	$ 0.00
49	0	$ 0.00	0	$ 0.00	1	$ 6079.81
50	2	$ 6600.00	2	$ 7608.00	4	$ 31158.26
51	1	$ 3336.00	1	$ 3966.00	0	$ 0.00
52	1	$ 3300.00	1	$ 3912.00	0	$ 0.00
53	3	$ 11166.01	3	$ 10506.00	4	$ 31158.26
54	0	$ 0.00	0	$ 0.00	2	$ 15662.83
55	1	$ 3300.00	1	$ 3804.00	3	$ 16381.59
56	1	$ 3300.00	1	$ 3858.00	0	$ 0.00
57	1	$ 3300.00	1	$ 3804.00	2	$ 15579.13
58	2	$ 7305.38	2	$ 7662.00	4	$ 31158.26
59	1	$ 3300.00	1	$ 3804.00	0	$ 0.00
60	0	$ 0.00	2	$ 7608.00	3	$ 21658.94
61	2	$ 6600.00	2	$ 7608.00	2	$ 15579.13
62	2	$ 7342.00	2	$ 7716.00	1	$ 7523.26
63	1	$ 3300.00	1	$ 3804.00	3	$ 16381.59
64	1	$ 3731.93	1	$ 3858.00	2	$ 18467.81
65	1	$ 3300.00	1	$ 3804.00	4	$ 31158.26
66	1	$ 3386.29	1	$ 3804.00	0	$ 0.00
67	1	$ 3300.00	1	$ 3804.00	3	$ 21032.69
68	2	$ 6600.00	2	$ 7608.00	2	$ 15579.13
69	1	$ 3300.00	1	$ 3804.00	2	$ 15579.13
70	1	$ 3336.00	1	$ 3966.00	2	$ 15746.53
TOTAL	66	$ 246796.44	75	$ 290910.31	109	$ 804422.75

D.15 STREET REQUIREMENTS

S T R E E T R E Q U I R E M E N T S

	STREET LIGHT		STREET SIGN		TRAFFIC SIGN	
LINK	QUANTITY	COST	QUANTITY	COST	QUANTITY	COST
1	2	$ 3000.00	1	$ 150.00	0	$ 0.00
2	2	$ 3000.00	0	$ 0.00	1	$ 150.00
3	0	$ 0.00	0	$ 0.00	0	$ 0.00
4	0	$ 0.00	0	$ 0.00	0	$ 0.00
5	0	$ 0.00	0	$ 0.00	0	$ 0.00
6	0	$ 0.00	0	$ 0.00	0	$ 0.00
7	1	$ 1500.00	1	$ 150.00	0	$ 0.00
8	3	$ 4500.00	0	$ 0.00	0	$ 0.00
9	1	$ 1500.00	0	$ 0.00	0	$ 0.00
10	1	$ 1500.00	1	$ 150.00	1	$ 150.00
11	1	$ 1500.00	0	$ 0.00	1	$ 150.00
12	3	$ 4500.00	0	$ 0.00	1	$ 150.00
13	0	$ 0.00	0	$ 0.00	0	$ 0.00
14	3	$ 4500.00	0	$ 0.00	1	$ 150.00
15	2	$ 3000.00	1	$ 150.00	0	$ 0.00
16	2	$ 3000.00	0	$ 0.00	1	$ 150.00
17	2	$ 3000.00	1	$ 150.00	0	$ 0.00
18	1	$ 1500.00	1	$ 150.00	1	$ 150.00
19	1	$ 1500.00	1	$ 150.00	0	$ 0.00
20	2	$ 3000.00	0	$ 0.00	0	$ 0.00
21	1	$ 1500.00	1	$ 150.00	0	$ 0.00
22	1	$ 1500.00	0	$ 0.00	0	$ 0.00
23	2	$ 3000.00	1	$ 150.00	0	$ 0.00
24	1	$ 1500.00	1	$ 150.00	1	$ 150.00
25	2	$ 3000.00	1	$ 150.00	0	$ 0.00
26	0	$ 0.00	0	$ 0.00	0	$ 0.00
27	0	$ 0.00	0	$ 0.00	1	$ 150.00
28	1	$ 1500.00	0	$ 0.00	0	$ 0.00
29	3	$ 4500.00	0	$ 0.00	0	$ 0.00
30	1	$ 1500.00	0	$ 0.00	0	$ 0.00
31	2	$ 3000.00	1	$ 150.00	0	$ 0.00
32	0	$ 0.00	0	$ 0.00	0	$ 0.00
33	1	$ 1500.00	0	$ 0.00	1	$ 150.00
34	1	$ 1500.00	0	$ 0.00	0	$ 0.00
35	1	$ 1500.00	1	$ 150.00	0	$ 0.00
36	1	$ 1500.00	0	$ 0.00	0	$ 0.00
37	0	$ 0.00	0	$ 0.00	0	$ 0.00
38	2	$ 3000.00	0	$ 0.00	1	$ 150.00
39	1	$ 1500.00	0	$ 0.00	0	$ 0.00
40	1	$ 1500.00	0	$ 0.00	0	$ 0.00
41	1	$ 1500.00	1	$ 150.00	0	$ 0.00

42	2	$ 3000.00	0	$ 0.00	1	$ 150.00
43	1	$ 1500.00	1	$ 150.00	0	$ 0.00
44	1	$ 1500.00	1	$ 150.00	0	$ 0.00
45	1	$ 1500.00	1	$ 150.00	1	$ 150.00
46	1	$ 1500.00	0	$ 0.00	0	$ 0.00
47	1	$ 1500.00	0	$ 0.00	0	$ 0.00
48	1	$ 1500.00	1	$ 150.00	0	$ 0.00
49	0	$ 0.00	0	$ 0.00	0	$ 0.00
50	3	$ 4500.00	1	$ 150.00	0	$ 0.00
51	0	$ 0.00	0	$ 0.00	0	$ 0.00
52	1	$ 1500.00	1	$ 150.00	0	$ 0.00
53	4	$ 6000.00	0	$ 0.00	0	$ 0.00
54	1	$ 1500.00	0	$ 0.00	0	$ 0.00
55	1	$ 1500.00	0	$ 0.00	1	$ 150.00
56	1	$ 1500.00	1	$ 150.00	0	$ 0.00
57	2	$ 3000.00	0	$ 0.00	1	$ 150.00
58	2	$ 3000.00	1	$ 150.00	0	$ 0.00
59	1	$ 1500.00	1	$ 150.00	0	$ 0.00
60	2	$ 3000.00	1	$ 150.00	0	$ 0.00
61	2	$ 3000.00	2	$ 300.00	1	$ 150.00
62	2	$ 3000.00	2	$ 300.00	1	$ 150.00
63	1	$ 1500.00	0	$ 0.00	1	$ 150.00
64	2	$ 3000.00	1	$ 150.00	0	$ 0.00
65	1	$ 1500.00	0	$ 0.00	1	$ 150.00
66	1	$ 1500.00	0	$ 0.00	0	$ 0.00
67	2	$ 3000.00	1	$ 150.00	0	$ 0.00
68	3	$ 4500.00	1	$ 150.00	0	$ 0.00
69	2	$ 3000.00	0	$ 0.00	0	$ 0.00
70	1	$ 1500.00	0	$ 0.00	1	$ 150.00
TOTAL	93	$ 139500.00	31	$ 4650.00	20	$ 3000.00

D.16 PAVEMENT INFORMATION

P A V E M E N T I N F O R M A T I O N

	LOCAL STREET			COLLECTOR STREET			TOTAL		
LINK	LENGTH	AREA (sq.m)	COST	LENGTH	AREA (sq.m)	COST	LENGTH	AREA (sq.m)	COST
1	0.00m	0.00	$ 0.00	108.49m	3128.74	$ 93862.20	108.49m	3128.74	$ 93862.20
2	110.40m	1104.00	$ 27600.00	0.00m	0.00	$ 0.00	110.40m	1104.00	$ 27600.00
3	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00
4	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00
5	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00
6	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00
7	0.00m	0.00	$ 0.00	79.92m	1598.40	$ 47952.00	79.92m	1598.40	$ 47952.00
8	0.00m	0.00	$ 0.00	246.89m	4937.80	$ 148134.00	246.89m	4937.80	$ 148134.00
9	0.00m	0.00	$ 0.00	74.68m	1493.60	$ 44808.00	74.68m	1493.60	$ 44808.00
10	0.00m	0.00	$ 0.00	60.96m	1219.20	$ 36576.00	60.96m	1219.20	$ 36576.00
11	66.20m	1324.00	$ 33100.00	0.00m	0.00	$ 0.00	66.20m	1324.00	$ 33100.00
12	236.89m	4737.80	$ 118444.99	0.00m	0.00	$ 0.00	236.89m	4737.80	$ 118444.99
13	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00
14	357.41m	1726.27	$ 43156.70	0.00m	0.00	$ 0.00	357.41m	1726.27	$ 43156.70
15	0.00m	0.00	$ 0.00	80.77m	1615.40	$ 48462.00	80.77m	1615.40	$ 48462.00
16	157.64m	1576.40	$ 39410.00	0.00m	0.00	$ 0.00	157.64m	1576.40	$ 39410.00
17	0.00m	0.00	$ 0.00	110.40m	2308.00	$ 69240.00	110.40m	2308.00	$ 69240.00
18	0.00m	0.00	$ 0.00	80.77m	1615.40	$ 48462.00	80.77m	1615.40	$ 48462.00
19	0.00m	0.00	$ 0.00	71.63m	1432.60	$ 42978.00	71.63m	1432.60	$ 42978.00
20	0.00m	0.00	$ 0.00	160.02m	3200.40	$ 96012.01	160.02m	3200.40	$ 96012.01
21	0.00m	0.00	$ 0.00	54.86m	1097.20	$ 32916.00	54.86m	1097.20	$ 32916.00
22	0.00m	0.00	$ 0.00	48.77m	975.40	$ 29262.00	48.77m	975.40	$ 29262.00
23	0.00m	0.00	$ 0.00	124.11m	3226.86	$ 96805.80	124.11m	3226.86	$ 96805.80
24	0.00m	0.00	$ 0.00	45.72m	914.40	$ 27432.00	45.72m	914.40	$ 27432.00
25	0.00m	0.00	$ 0.00	70.10m	1402.00	$ 42060.00	70.10m	1402.00	$ 42060.00
26	0.00m	0.00	$ 0.00	38.10m	762.00	$ 22860.00	38.10m	762.00	$ 22860.00
27	63.82m	1276.40	$ 31910.00	0.00m	0.00	$ 0.00	63.82m	1276.40	$ 31910.00
28	73.15m	1463.00	$ 36575.00	0.00m	0.00	$ 0.00	73.15m	1463.00	$ 36575.00
29	123.44m	2470.13	$ 61753.29	0.00m	0.00	$ 0.00	123.44m	2470.13	$ 61753.29
30	80.77m	1615.40	$ 40385.00	0.00m	0.00	$ 0.00	80.77m	1615.40	$ 40385.00
31	152.40m	3048.00	$ 76200.00	0.00m	0.00	$ 0.00	152.40m	3048.00	$ 76200.00
32	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00
33	81.56m	1631.20	$ 40780.00	0.00m	0.00	$ 0.00	81.56m	1631.20	$ 40780.00
34	76.14m	853.88	$ 21347.07	0.00m	0.00	$ 0.00	76.14m	853.88	$ 21347.07
35	77.30m	1546.00	$ 38650.00	0.00m	0.00	$ 0.00	77.30m	1546.00	$ 38650.00
36	49.44m	494.40	$ 12360.00	0.00m	0.00	$ 0.00	49.44m	494.40	$ 12360.00
37	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00	0.00m	0.00	$ 0.00
38	159.16m	3183.20	$ 79580.01	0.00m	0.00	$ 0.00	159.16m	3183.20	$ 79580.01
39	59.44m	1188.80	$ 29720.00	0.00m	0.00	$ 0.00	59.44m	1188.80	$ 29720.00
40	57.91m	1158.20	$ 28955.00	0.00m	0.00	$ 0.00	57.91m	1158.20	$ 28955.00
41	82.30m	1646.00	$ 41150.00	0.00m	0.00	$ 0.00	82.30m	1646.00	$ 41150.00

42	164.83m	3296.60	$	82415.00	0.00m	0.00	$	0.00	164.83m	3296.60	$	82415.00
43	0.00m	0.00	$	0.00	82.30m	1646.00	$	49380.00	82.30m	1646.00	$	49380.00
44	0.00m	0.00	$	0.00	76.92m	2077.92	$	62337.60	76.92m	2077.92	$	62337.60
45	0.00m	0.00	$	0.00	83.82m	1676.40	$	50292.00	83.82m	1676.40	$	50292.00
46	0.00m	0.00	$	0.00	80.77m	1615.40	$	48462.00	80.77m	1615.40	$	48462.00
47	52.91m	579.10	$	14477.50	0.00m	0.00	$	0.00	52.91m	579.10	$	14477.50
48	0.00m	0.00	$	0.00	54.86m	1097.20	$	32916.00	54.86m	1097.20	$	32916.00
49	56.20m	1124.00	$	28100.00	0.00m	0.00	$	0.00	56.20m	1124.00	$	28100.00
50	196.60m	3932.00	$	98300.01	0.00m	0.00	$	0.00	196.60m	3932.00	$	98300.01
51	0.00m	0.00	$	0.00	0.00m	0.00	$	0.00	0.00m	0.00	$	0.00
52	0.00m	0.00	$	0.00	39.62m	792.40	$	23772.00	39.62m	792.40	$	23772.00
53	0.00m	0.00	$	0.00	264.37m	5287.40	$	158622.00	264.37m	5287.40	$	158622.00
54	0.00m	0.00	$	0.00	132.64m	2712.80	$	81384.00	132.64m	2712.80	$	81384.00
55	157.81m	910.62	$	22765.60	0.00m	0.00	$	0.00	157.81m	910.62	$	22765.60
56	0.00m	0.00	$	0.00	100.58m	2011.60	$	60348.00	100.58m	2011.60	$	60348.00
57	154.59m	3091.80	$	77295.00	0.00m	0.00	$	0.00	154.59m	3091.80	$	77295.00
58	90.34m	1806.80	$	45170.00	0.00m	0.00	$	0.00	90.34m	1806.80	$	45170.00
59	0.00m	0.00	$	0.00	57.91m	1158.20	$	34746.00	57.91m	1158.20	$	34746.00
60	0.00m	0.00	$	0.00	160.69m	3213.80	$	96414.00	160.69m	3213.80	$	96414.00
61	0.00m	0.00	$	0.00	94.49m	1889.80	$	56694.00	94.49m	1889.80	$	56694.00
62	0.00m	0.00	$	0.00	91.44m	2377.44	$	71323.20	91.44m	2377.44	$	71323.20
63	57.06m	570.60	$	14265.00	0.00m	0.00	$	0.00	57.06m	570.60	$	14265.00
64	0.00m	0.00	$	0.00	118.92m	3169.92	$	95097.60	118.92m	3169.92	$	95097.60
65	152.26m	3045.20	$	76130.00	0.00m	0.00	$	0.00	152.26m	3045.20	$	76130.00
66	104.57m	1012.50	$	25312.57	0.00m	0.00	$	0.00	104.57m	1012.50	$	25312.57
67	86.87m	1737.40	$	43435.00	0.00m	0.00	$	0.00	86.87m	1737.40	$	43435.00
68	186.78m	3735.60	$	93390.00	0.00m	0.00	$	0.00	186.78m	3735.60	$	93390.00
69	103.63m	2072.60	$	51815.00	0.00m	0.00	$	0.00	103.63m	2072.60	$	51815.00
70	104.30m	2086.00	$	52150.00	0.00m	0.00	$	0.00	104.30m	2086.00	$	52150.00
TOTAL	3629.12m	61043.92	$	1526097.75	2515.52m	61653.68	$	1849610.42	6144.64m	122697.60	$	3375708.17

D.17 SIDEWALK REQUIREMENTS

S I D E W A L K R E Q U I R E M E N T S

LINK	BARRIER CURB/GUTTER LENGTH	ROLLED CURB/GUTTER LENGTH	BLVD. CURB/GUTTER LENGTH	SIDEWALK AREA (sq. m)	COST
1	140.20m	106.70m	0.00m	203.46	$ 13560.20
2	6.10m	262.10m	0.00m	288.95	$ 17056.16
3	0.00m	0.00m	9.10m	0.00	$ 136.50
4	0.00m	0.00m	0.00m	0.00	$ 0.00
5	0.00m	0.00m	0.00m	0.00	$ 0.00
6	0.00m	0.00m	0.00m	0.00	$ 0.00
7	91.40m	67.10m	0.00m	98.76	$ 7278.70
8	6.10m	472.40m	0.00m	215.80	$ 16918.82
9	3.00m	134.10m	0.00m	89.62	$ 6104.22
10	50.30m	48.80m	0.00m	73.15	$ 5029.84
11	6.10m	125.00m	0.00m	76.81	$ 5453.54
12	103.60m	373.40m	0.00m	294.43	$ 20922.44
13	0.00m	0.00m	0.00m	0.00	$ 0.00
14	6.10m	353.60m	9.10m	416.96	$ 24325.88
15	9.10m	125.00m	0.00m	96.92	$ 6418.58
16	6.10m	353.60m	9.10m	416.96	$ 24325.88
17	6.10m	217.90m	0.00m	169.17	$ 11003.15
18	9.10m	125.00m	0.00m	98.76	$ 6501.20
19	88.40m	54.90m	0.00m	85.96	$ 6459.52
20	24.40m	410.00m	0.00m	265.18	$ 18570.92
21	3.00m	105.20m	0.00m	67.67	$ 4683.06
22	3.00m	82.30m	0.00m	58.52	$ 3928.08
23	6.10m	243.80m	0.00m	182.88	$ 12008.60
24	6.10m	61.00m	0.00m	54.86	$ 3505.88
25	3.00m	126.50m	0.00m	82.30	$ 5660.82
26	3.00m	59.40m	0.00m	45.72	$ 3008.40
27	9.10m	135.60m	0.00m	89.62	$ 6248.72
28	6.10m	126.50m	0.00m	78.64	$ 5558.12
29	3.00m	245.40m	0.00m	149.96	$ 10489.38
30	6.10m	158.50m	0.00m	98.76	$ 6943.70
31	6.10m	286.50m	0.00m	181.06	$ 12567.02
32	0.00m	0.00m	0.00m	0.00	$ 0.00
33	9.10m	153.90m	0.00m	106.07	$ 7263.56
34	6.10m	64.00m	7.60m	64.01	$ 4076.36
35	6.10m	131.10m	0.00m	98.76	$ 6532.70
36	3.00m	129.50m	9.10m	179.22	$ 10203.90
37	0.00m	0.00m	0.00m	0.00	$ 0.00
38	9.10m	306.30m	0.00m	193.85	$ 13499.66
39	3.00m	120.40m	0.00m	71.33	$ 5075.76
40	3.00m	109.70m	0.00m	69.49	$ 4832.64
41	91.40m	59.40m	0.00m	96.92	$ 7080.58

42	15.20m	164.60m	0.00m	208.49	$	12154.96
43	9.10m	128.00m	0.00m	87.78	$	6052.10
44	3.00m	169.20m	0.00m	150.87	$	9387.15
45	3.00m	155.50m	0.00m	87.78	$	6342.60
46	3.00m	149.40m	0.00m	98.76	$	6745.20
47	6.10m	134.30m	9.10m	149.96	$	9021.38
48	6.10m	82.30m	0.00m	65.83	$	4318.94
49	12.20m	109.70m	0.00m	73.15	$	5181.34
50	3.00m	376.40m	0.00m	235.92	$	16322.40
51	0.00m	0.00m	0.00m	0.00	$	0.00
52	3.00m	67.10m	0.00m	45.72	$	3123.90
53	187.50m	356.60m	0.00m	334.67	$	24159.06
54	6.10m	253.00m	0.00m	162.77	$	11241.36
55	6.10m	163.10m	9.10m	153.62	$	9618.08
56	6.10m	179.80m	0.00m	120.70	$	8250.32
57	9.10m	297.20m	0.00m	184.70	$	12951.68
58	9.10m	149.40m	0.00m	93.26	$	6619.88
59	6.10m	56.40m	0.00m	69.49	$	4095.14
60	6.10m	321.60m	0.00m	244.61	$	15953.23
61	6.10m	163.10m	0.00m	113.39	$	7670.96
62	12.20m	128.00m	0.00m	98.76	$	6608.20
63	6.10m	157.00m	9.10m	153.62	$	9526.58
64	6.10m	214.90m	0.00m	180.60	$	11472.30
65	9.10m	306.30m	0.00m	195.68	$	13582.28
66	3.00m	112.80m	9.10m	106.07	$	6661.56
67	6.10m	160.00m	0.00m	106.07	$	7295.06
68	6.10m	341.40m	0.00m	213.97	$	14871.74
69	6.10m	199.60m	0.00m	124.36	$	8712.02
70	9.10m	211.80m	0.00m	128.02	$	9119.72
TOTAL	1111.40m	11243.10m	80.40m	8849.15	$	590292.13

D.18 FAMILY UNITS

A M I L Y U N I T S

	SINGLE FAMILY				MULTI-FAMILY				
LINK	UNITS	GAS	POWER	TELEPHONE	UNITS	GAS	POWER	TELEPHONE	TOTAL
1	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
2	14	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
3	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
4	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
5	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
6	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
7	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
8	32	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
9	0	$ 0.00	$ 0.00	$ 0.00	10	$ 0.00	$ 0.00	$ 0.00	$ 0.00
10	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
11	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
12	26	$ 0.00	$ 0.00	$ 0.00	35	$ 0.00	$ 0.00	$ 0.00	$ 0.00
13	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
14	30	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
15	0	$ 0.00	$ 0.00	$ 0.00	30	$ 0.00	$ 0.00	$ 0.00	$ 0.00
16	32	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
17	9	$ 0.00	$ 0.00	$ 0.00	37	$ 0.00	$ 0.00	$ 0.00	$ 0.00
18	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
19	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
20	34	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
21	4	$ 0.00	$ 0.00	$ 0.00	25	$ 0.00	$ 0.00	$ 0.00	$ 0.00
22	2	$ 0.00	$ 0.00	$ 0.00	13	$ 0.00	$ 0.00	$ 0.00	$ 0.00
23	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
24	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
25	15	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
26	3	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
27	9	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
28	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
29	27	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
30	7	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
31	25	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
32	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
33	7	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
34	6	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
35	7	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
36	6	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
37	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
38	19	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
39	6	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
40	8	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
41	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00

42	27	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
43	0	$ 0.00	$ 0.00	$ 0.00	26	$ 0.00	$ 0.00	$ 0.00	$ 0.00
44	0	$ 0.00	$ 0.00	$ 0.00	125	$ 0.00	$ 0.00	$ 0.00	$ 0.00
45	7	$ 0.00	$ 0.00	$ 0.00	23	$ 0.00	$ 0.00	$ 0.00	$ 0.00
46	2	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
47	14	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
48	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
49	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
50	33	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
51	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
52	1	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
53	26	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
54	19	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
55	13	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
56	0	$ 0.00	$ 0.00	$ 0.00	20	$ 0.00	$ 0.00	$ 0.00	$ 0.00
57	23	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
58	16	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
59	0	$ 0.00	$ 0.00	$ 0.00	20	$ 0.00	$ 0.00	$ 0.00	$ 0.00
60	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
61	6	$ 0.00	$ 0.00	$ 0.00	24	$ 0.00	$ 0.00	$ 0.00	$ 0.00
62	0	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
63	13	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
64	11	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
65	29	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
66	11	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
67	12	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
68	31	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
69	13	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
70	18	$ 0.00	$ 0.00	$ 0.00	0	$ 0.00	$ 0.00	$ 0.00	$ 0.00
TOTAL	653	$ 0.00	$ 0.00	$ 0.00	388	$ 0.00	$ 0.00	$ 0.00	$ 0.00

D.19 CONSTRUCTION COSTS

CONSTRUCTION COST

LINK	CONSTRUCTION COST	CONSTRUCTION COST PER PAVEMENT AREA	CONSTRUCTION COST PER STREET LENGTH	CONSTRUCTION COST PER SALEABLE FRONTAGE	CONSTRUCTION COST PER DWELLING UNIT
1	$ 212828.70	68.02 $/sq. m	1961.74 $/m	0.00 $/m	0.00 $/units
2	$ 171106.11	154.99 $/sq. m	1549.87 $/m	664.36 $/m	12221.87 $/units
3	$ 133133.81	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
4	$ 98511.36	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
5	$ 56713.35	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
6	$ 53617.27	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
7	$ 136483.19	85.39 $/sq. m	1707.75 $/m	0.00 $/m	0.00 $/units
8	$ 434460.09	87.99 $/sq. m	1759.73 $/m	934.68 $/m	13576.88 $/units
9	$ 130068.13	87.08 $/sq. m	1741.67 $/m	0.00 $/m	13006.81 $/units
10	$ 95177.34	78.07 $/sq. m	1561.31 $/m	0.00 $/m	0.00 $/units
11	$ 86533.14	65.36 $/sq. m	1307.15 $/m	0.00 $/m	0.00 $/units
12	$ 487203.13	102.83 $/sq. m	2056.66 $/m	1507.95 $/m	7986.94 $/units
13	$ 93607.40	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
14	$ 345778.28	200.30 $/sq. m	967.46 $/m	1064.26 $/m	11525.94 $/units
15	$ 172256.19	106.63 $/sq. m	2132.68 $/m	0.00 $/m	5741.87 $/units
16	$ 262297.03	166.39 $/sq. m	1663.90 $/m	807.32 $/m	8196.78 $/units
17	$ 234827.28	101.74 $/sq. m	2127.06 $/m	2266.02 $/m	5104.94 $/units
18	$ 121200.27	75.03 $/sq. m	1500.56 $/m	0.00 $/m	0.00 $/units
19	$ 119330.05	83.30 $/sq. m	1665.92 $/m	0.00 $/m	0.00 $/units
20	$ 238116.23	74.40 $/sq. m	1488.04 $/m	504.01 $/m	7003.42 $/units
21	$ 93358.59	85.09 $/sq. m	1701.76 $/m	0.00 $/m	3219.26 $/units
22	$ 69365.49	71.11 $/sq. m	1422.30 $/m	2845.18 $/m	4624.37 $/units
23	$ 208141.45	64.50 $/sq. m	1677.07 $/m	0.00 $/m	0.00 $/units
24	$ 81061.78	88.65 $/sq. m	1773.00 $/m	0.00 $/m	0.00 $/units
25	$ 124745.93	88.98 $/sq. m	1779.54 $/m	772.23 $/m	8316.40 $/units
26	$ 83581.33	109.69 $/sq. m	2193.74 $/m	0.00 $/m	27860.44 $/units
27	$ 135714.72	106.33 $/sq. m	2126.52 $/m	947.33 $/m	15079.41 $/units
28	$ 112979.32	77.22 $/sq. m	1544.49 $/m	1219.29 $/m	0.00 $/units
29	$ 219964.08	89.05 $/sq. m	1781.95 $/m	874.75 $/m	8146.82 $/units
30	$ 132483.91	82.01 $/sq. m	1640.26 $/m	1358.25 $/m	18926.27 $/units
31	$ 255201.03	83.73 $/sq. m	1674.55 $/m	669.82 $/m	10208.04 $/units
32	$ 69067.68	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
33	$ 146710.45	89.94 $/sq. m	1798.80 $/m	1631.57 $/m	20958.64 $/units
34	$ 73482.70	86.06 $/sq. m	965.08 $/m	861.06 $/m	12247.12 $/units
35	$ 137071.59	88.66 $/sq. m	1773.24 $/m	1798.84 $/m	19581.66 $/units
36	$ 94128.45	190.39 $/sq. m	1903.89 $/m	1211.12 $/m	15688.07 $/units
37	$ 69466.76	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
38	$ 316681.97	99.49 $/sq. m	1989.71 $/m	1323.53 $/m	16667.47 $/units
39	$ 109445.05	92.06 $/sq. m	1841.27 $/m	2244.11 $/m	18240.84 $/units
40	$ 112405.72	97.05 $/sq. m	1941.04 $/m	899.46 $/m	14050.71 $/units
41	$ 114126.98	69.34 $/sq. m	1386.72 $/m	3119.93 $/m	0.00 $/units

42	$ 301066.19	91.33 $/sq. m	1826.53 $/m	914.59 $/m	11150.60 $/units
43	$ 145817.31	88.59 $/sq. m	1771.78 $/m	0.00 $/m	5608.36 $/units
44	$ 185006.23	89.03 $/sq. m	2405.18 $/m	0.00 $/m	1480.05 $/unit
45	$ 175739.13	104.83 $/sq. m	2096.63 $/m	1961.16 $/m	5857.97 $/units
46	$ 146035.06	90.40 $/sq. m	1808.04 $/m	3523.16 $/m	73017.53 $/units
47	$ 90759.87	156.79 $/sq. m	1716.12 $/m	744.75 $/m	6485.70 $/units
48	$ 101232.85	92.31 $/sq. m	1846.21 $/m	0.00 $/m	0.00 $/units
49	$ 85690.75	76.24 $/sq. m	1524.75 $/m	395.96 $/m	0.00 $/units
50	$ 350032.50	89.02 $/sq. m	1780.43 $/m	841.32 $/m	10607.05 $/units
51	$ 49033.26	0.00 $/sq. m	0.00 $/m	0.00 $/m	0.00 $/units
52	$ 72129.05	91.03 $/sq. m	1820.52 $/m	0.00 $/m	72129.05 $/units
53	$ 504558.09	95.43 $/sq. m	1908.53 $/m	1263.64 $/m	19406.08 $/units
54	$ 237018.72	87.37 $/sq. m	1786.93 $/m	925.75 $/m	12474.67 $/units
55	$ 116280.45	127.69 $/sq. m	736.82 $/m	578.02 $/m	8944.65 $/units
56	$ 169738.78	84.38 $/sq. m	1687.60 $/m	0.00 $/m	8486.94 $/units
57	$ 267173.41	86.41 $/sq. m	1728.27 $/m	963.24 $/m	11616.24 $/units
58	$ 179407.64	99.30 $/sq. m	1985.92 $/m	957.10 $/m	11212.98 $/units
59	$ 102204.28	88.24 $/sq. m	1764.88 $/m	0.00 $/m	5110.21 $/units
60	$ 252830.92	78.67 $/sq. m	1573.41 $/m	0.00 $/m	0.00 $/units
61	$ 184343.91	97.55 $/sq. m	1950.94 $/m	2050.09 $/m	6144.80 $/units
62	$ 159558.20	67.11 $/sq. m	1744.95 $/m	0.00 $/m	0.00 $/units
63	$ 110488.24	193.64 $/sq. m	1936.35 $/m	557.68 $/m	8499.10 $/units
64	$ 251479.94	79.33 $/sq. m	2114.70 $/m	1602.09 $/m	22861.81 $/units
65	$ 290513.25	95.40 $/sq. m	1908.01 $/m	1887.43 $/m	10017.70 $/units
66	$ 71444.97	70.56 $/sq. m	683.23 $/m	558.08 $/m	6495.00 $/units
67	$ 161763.42	93.11 $/sq. m	1862.13 $/m	1326.80 $/m	13480.29 $/units
68	$ 304982.94	81.64 $/sq. m	1632.85 $/m	593.83 $/m	9838.16 $/units
69	$ 181342.48	87.50 $/sq. m	1749.90 $/m	1565.73 $/m	13949.42 $/units
70	$ 192267.25	92.17 $/sq. m	1843.41 $/m	888.44 $/m	10681.51 $/units
TOTAL	$ 11882461.00				

D.20 SUMMARY-CONSTRUCTION COSTS

S U M M A R Y

DESCRIPTION	QUANTITY		COST	
WATERMAIN	6490.73m		$ 957196.44	
DOMESTIC SEWER	5823.21m		$ 1900113.13	
STORM SEWER	7193.29m		$ 3569873.00	
COLLECTOR STREET	61653.68sq. m		$ 1849610.42	
LOCAL STREET	61043.92sq. m		$ 1526097.75	
SUBTOTAL PAVEMENT		122697.60sq. m		$ 3375708.17
SIDEWALK	8849.15sq. m		$ 398211.59	
BARRIER CURB & GUTTER	1111.40m		$ 22227.99	
ROLLED CURB & GUTTER	11243.10m		$ 168646.50	
BLVD. CURB & GUTTER	80.40m		$ 1206.00	
SUBTOTAL SIDEWALK REQUIREMENT				$ 590292.13
PARK DEVELOPMENT	7.37Ha		$ 368500.00	
BUFFER DEVELOPMENT	0.28Ha		$ 14000.00	
CATCH-BASIN	109 units		$ 804422.75	
DOMESTIC SEWER MANHOLE	66 units		$ 246796.44	
STORM SEWER MANHOLE	75 units		$ 290910.31	
STREET LIGHT	93 units		$ 139500.00	
STREET SIGN	31 units		$ 4650.00	
TRAFFIC SIGN	20 units		$ 3000.00	
SINGLE FAMILY	653 units			
- GAS			$ 0.00	
- POWER			$ 0.00	
- TELEPHONE			$ 0.00	
MULTI-FAMILY	388 units			
- GAS			$ 0.00	
- POWER			$ 0.00	
- TELEPHONE			$ 0.00	
SUBTOTAL UTILITIES		1041 units		$ 0.00
T O T A L			$ 12264964.00	

D.21 SITE SPECIFIC INFORMATION

S I T E S P E C I F I C I N F O R M A T I O N

Road Elements :

DESCRIPTION	QUANTITY
Cul-de-sacs	10
Dead-ends	0
Three-way intersections	22
Four-way intersections	2
Subdivision exits	7

Land Use Distribution :

DESCRIPTION	AREA
Net single family	21.60 Ha
Net multi-family	12.00 Ha
School	7.50 Ha
Park	7.37 Ha
Buffer	0.28 Ha
Commercial	13.10 Ha
Pavement	12.27 Ha
Sidewalk	0.88 Ha
Other	1.90 Ha
Gross Subdivision Area	76.90 Ha

```
el
.page
.header level1 ^%SUBDIVISION CONSTRUCTION COSTS\%
.lt
```

SUBDIVISION CONSTRUCTION COST

Total construction cost per total subdivision area = 159492.38 $/Ha

gross area	=	198141.58 $/Ha
net area	=	365079.47 $/Ha
single family area	=	567945.31 $/Ha
pavement area	=	999609.13 $/Ha
street length	=	1996.04 $/m
saleable frontage	=	1032.05 $/m
total dwelling unit	=	11781.91 $/units
single family	=	18782.49 $/units

D.22 SUBDIVISION DENSITY

S U B D I V I S I O N D E N S I T Y

Unit density based on GROSS area :

FAMILY UNIT	QUANTITY	GROSS AREA	DENSITY
Single	653 units	49.90 Ha	13.09 units/Ha
Multi	388 units	40.30 Ha	9.63 units/Ha
Total	1041 units	61.90 Ha	16.82 units/Ha

Unit density based on NET area :

FAMILY UNIT	QUANTITY	NET AREA	DENSITY
Single	653 units	33.60 Ha	19.44 units/Ha
Multi	388 units	33.60 Ha	11.55 units/Ha
Total	1041 units	33.60 Ha	30.99 units/Ha

D.23 YIELD EFFICIENCY

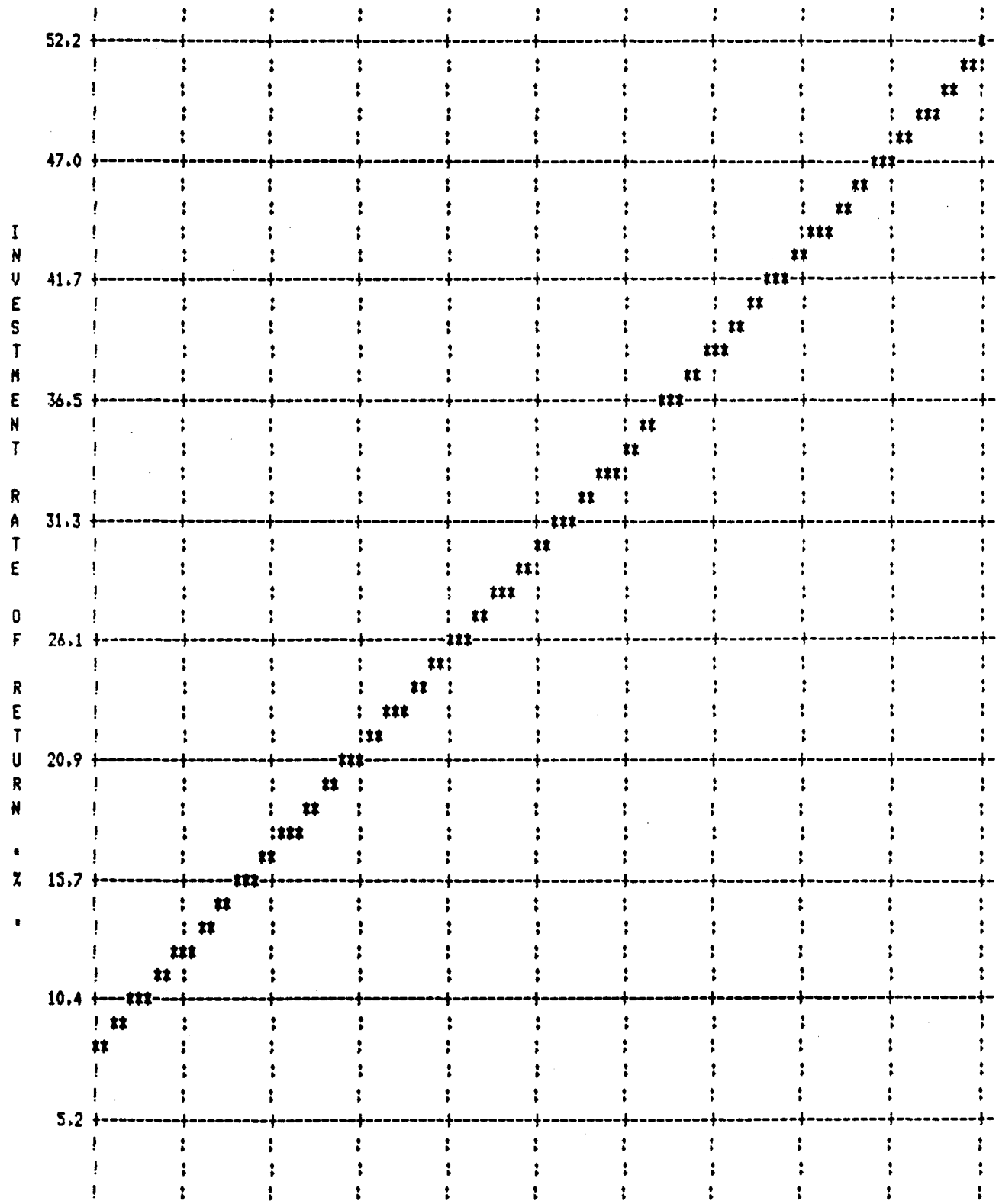
YIELD EFFICIENCY

Saleable frontage :	11884.06m
Street length :	6144.64m
Gross subdivision area :	61.90 Ha

YIELD - based on Total Street Length :	1.93 m/m
- based on Gross Subdivision Area :	191.99 m/Ha

D.24 INVESTMENT RATE OF RETURN GRAPH

INVESTMENT RATE OF RETURN versus REVENUE



```
     !         :         :         :         :         :         :         :         :         :         :
0.0 L.........:.........:.........:.........:.........:.........:.........:.........:.........:.........:.
   2500.     2600.     2700.     2800.     2900.     3000.     3100.     3200.     3300.     3400.     3500.

                                        REVENUE ( $ per meter )
```

IRR = 0.4348E-03 x (REVENUE) - 1

.5 MAINTENANCE COSTS

M A I N T E N A N C E C O S T

DESCRIPTION	QUANTITY		UNIT COST	MAINTENANCE COST
WATERMAIN	6490.73m		$ 11.00	$ 71398.03
DOMESTIC SEWER	5823.21m		$ 13.00	$ 75701.73
STORM SEWER	7193.29m		$ 10.00	$ 71932.89
PAVEMENT	122697.60sq. m		$ 15.00	$ 1840464.00
SIDEWALK	8849.15sq. m		$ 12.00	$ 106189.76
BARRIER CURB & GUTTER	1111.40m		$ 11.00	$ 12225.39
ROLLED CURB & GUTTER	11243.10m		$ 13.00	$ 146160.30
BLVD. CURB & GUTTER	80.40m		$ 10.00	$ 804.00
PARK DEVELOPMENT	7.37Ha		$ 15.00	$ 110.55
BUFFER DEVELOPMENT	0.28Ha		$ 12.00	$ 3.36
CATCH-BASIN	109	units	$ 11.00	$ 1199.00
MANHOLE	141	units	$ 13.00	$ 1833.00
STREET LIGHT	93	units	$ 10.00	$ 930.00
STREET SIGN	31	units	$ 15.00	$ 465.00
TRAFFIC SIGN	20	units	$ 12.00	$ 240.00
SINGLE FAMILY	653	units		
- GAS			$ 11.00	$ 7183.00
- POWER			$ 13.00	$ 8489.00
- TELEPHONE			$ 10.00	$ 6530.00
MULTI-FAMILY	388	units		
- GAS			$ 15.00	$ 5820.00
- POWER			$ 12.00	$ 4656.00
- TELEPHONE			$ 11.00	$ 4268.00
T O T A L				$ 2366603.00

26 OPERATION COSTS

O P E R A T I O N C O S T

DESCRIPTION	QUANTITY	UNIT COST	OPERATION COST
WATERMAIN	6490.73m	$ 0.85	$ 5517.12
DOMESTIC SEWER	5823.21m	$ 1.00	$ 5823.21
STORM SEWER	7193.29m	$ 1.00	$ 7193.29
PAVEMENT	122697.60sq. m	$ 0.87	$ 106746.91
SIDEWALK	8849.15sq. m	$ 0.95	$ 8406.69
BARRIER CURB & GUTTER	1111.40m	$ 0.85	$ 944.69
ROLLED CURB & GUTTER	11243.10m	$ 1.00	$ 11243.10
BLVD. CURB & GUTTER	80.40m	$ 1.00	$ 80.40
PARK DEVELOPMENT	7.37Ha	$ 0.87	$ 6.41
BUFFER DEVELOPMENT	0.28Ha	$ 0.95	$ 0.27
CATCH-BASIN	109 units	$ 0.85	$ 92.65
MANHOLE	141 units	$ 1.00	$ 141.00
STREET LIGHT	93 units	$ 1.00	$ 93.00
STREET SIGN	31 units	$ 0.87	$ 26.97
TRAFFIC SIGN	20 units	$ 0.95	$ 19.00
SINGLE FAMILY	653 units		
- GAS		$ 0.85	$ 555.05
- POWER		$ 1.00	$ 653.00
- TELEPHONE		$ 1.00	$ 653.00
MULTI-FAMILY	388 units		
- GAS		$ 0.87	$ 337.56
- POWER		$ 0.95	$ 368.60
- TELEPHONE		$ 0.85	$ 329.80
T O T A L			$ 149231.70

.7 DETAILED ANALYSIS

D E T A I L E D A N A L Y S I S

CONSTRUCTION COST : $ 12264964.00

ANNUAL MAINTENANCE COST : $ 2388494.25

ANNUAL OPERATION COST : $ 222966.05

INTEREST	LIFE	PRESENT WORTH	ANNUAL COST
5.00 %	1 yr	$ 14752067.00	$ 15489685.00
5.00 %	10 yr	$ 32429948.00	$ 4199831.00
5.00 %	20 yr	$ 44809512.00	$ 3595633.50
5.00 %	50 yr	$ 59939576.00	$ 3283295.25
5.00 %	100 yr	$ 64096988.00	$ 3229407.75
13.50 %	10 yr	$ 26156670.00	$ 4917114.00
13.50 %	20 yr	$ 30072260.00	$ 4410131.50
13.50 %	50 yr	$ 31574698.00	$ 4270181.50
13.50 %	100 yr	$ 31609052.00	$ 4267236.00
13.50 %	INFINITE	$ 31609114.00	$ 4267230.50

28 CONSTRUCTION COST BREAKDOWN

C O N S T R U C T I O N C O S T B R E A K D O W N

INTEREST RATE : 13.50 %
SUBDIVISION's LIFE : 100 yrs.

DESCRIPTION	ACCOUNT NUMBER	PRESENT COST	ANNUAL COST
WATERMAIN			
	234-345-4561	$ 239299.11	$ 32305.48
	567-678-6789	$ 717897.31	$ 96916.45
T O T A L		$ 957196.44	$ 129221.94
DOMESTIC SEWER			
	123-456-7897	$ 969057.69	$ 130823.21
	567-678-6789	$ 931055.44	$ 125692.89
T O T A L		$ 1900113.13	$ 256516.11
STORM SEWER			
	123-456-7897	$ 0.00	$ 0.00
	234-345-4561	$ 0.00	$ 0.00
	567-678-6789	$ 0.00	$ 0.00
	132-354-4657	$ 0.00	$ 0.00
T O T A L		$ 3569873.00	$ 0.00
PAVEMENT			
	123-456-7897	$ 1181497.75	$ 159502.72
	567-678-6789	$ 843927.00	$ 113930.52
	756-143-6472	$ 1350283.25	$ 182288.83
T O T A L		$ 3375708.00	$ 455722.09
SIDEWALK			
	234-345-4561	$ 274766.00	$ 37093.53
	567-678-6789	$ 123445.59	$ 16665.21
T O T A L		$ 398211.59	$ 53758.74
BARRIER CURB & GUTTER			
	132-354-4657	$ 22227.99	$ 3000.79
T O T A L		$ 22227.99	$ 3000.79
ROLLED CURB & GUTTER			
	132-354-4657	$ 84323.25	$ 11383.68
	756-143-6472	$ 84323.25	$ 11383.68

	T O T A L		$ 168646.50	$ 22767.35
BLVD. CURB & GUTTER				
		234-345-4561	$ 120.60	$ 16.28
		567-678-6789	$ 241.20	$ 32.56
		756-143-6472	$ 844.20	$ 113.97
	T O T A L		$ 1206.00	$ 162.81
PARK DEVELOPMENT				
		132-354-4657	$ 22227.99	$ 3000.79
	T O T A L		$ 368500.00	$ 3000.79
BUFFER DEVELOPMENT				
		123-456-7897	$ 2800.00	$ 378.00
		234-345-4561	$ 4200.00	$ 567.00
		567-678-6789	$ 5600.00	$ 756.00
		132-354-4657	$ 700.00	$ 94.50
		756-143-6472	$ 700.00	$ 94.50
	T O T A L		$ 14000.00	$ 1890.01
CATCH-BASIN				
		123-456-7897	$ 80442.27	$ 10859.74
		756-143-6472	$ 723980.44	$ 97737.68
	T O T A L		$ 804422.75	$ 108597.42
MANHOLE				
		234-345-4561	$ 537706.69	$ 72590.64
	T O T A L		$ 537706.69	$ 72590.64
STREET LIGHT				
		123-456-7897	$ 41850.00	$ 5649.77
		234-345-4561	$ 34875.00	$ 4708.14
		567-678-6789	$ 27900.00	$ 3766.51
		756-143-6472	$ 34875.00	$ 4708.14
	T O T A L		$ 139500.00	$ 18832.56
STREET SIGN				
		234-345-4561	$ 2325.00	$ 313.88
		567-678-6789	$ 2325.00	$ 313.88
	T O T A L		$ 4650.00	$ 627.75
TRAFFIC SIGN				
		756-143-6472	$ 3000.00	$ 405.00
	T O T A L		$ 3000.00	$ 405.00
SINGLE FAMILY				
- GAS				
		567-678-6789	$ 0.00	$ 0.00
		132-354-4657	$ 0.00	$ 0.00

	756-143-6472	$	0.00	$	0.00
T O T A L		$	0.00	$	0.00
- POWER					
	234-345-4561	$	0.00	$	0.00
	132-354-4657	$	0.00	$	0.00
T O T A L		$	0.00	$	0.00
- TELEPHONE					
	123-456-7897	$	0.00	$	0.00
	567-678-6789	$	0.00	$	0.00
	132-354-4657	$	0.00	$	0.00
	756-143-6472	$	0.00	$	0.00
T O T A L		$	0.00	$	0.00
MULTI-FAMILY					
- GAS					
	567-678-6789	$	0.00	$	0.00
T O T A L		$	0.00	$	0.00
- POWER					
	123-456-7897	$	0.00	$	0.00
	234-345-4561	$	0.00	$	0.00
	567-678-6789	$	0.00	$	0.00
	132-354-4657	$	0.00	$	0.00
	756-143-6472	$	0.00	$	0.00
T O T A L		$	0.00	$	0.00
- TELEPHONE					
	756-143-6472	$	0.00	$	0.00
T O T A L		$	0.00	$	0.00
T O T A L		$	12264964.00	$	1655775.50

29 MAINTENANCE COST BREAKDOWN

MAINTENANCE COST BREAKDOWN

DESCRIPTION		ACCOUNT NUMBER	PRESENT COST	ANNUAL COST
WATERMAIN				
INTEREST RATE :	8.00 %			
SERVICE LIFE :	4 yrs.			
		234-345-4561	$ 59586.04	$ 17990.26
		567-678-6789	$ 178758.11	$ 53970.77
TOTAL			$ 238344.14	$ 71961.02
DOMESTIC SEWER				
INTEREST RATE :	6.00 %			
SERVICE LIFE :	6 yrs.			
		123-456-7897	$ 192009.69	$ 39047.63
		567-678-6789	$ 184479.91	$ 37516.34
TOTAL			$ 376489.59	$ 76563.97
STORM SEWER				
INTEREST RATE :	3.00 %			
SERVICE LIFE :	8 yrs.			
		123-456-7897	$ 0.00	$ 0.00
		234-345-4561	$ 0.00	$ 0.00
		567-678-6789	$ 0.00	$ 0.00
		132-354-4657	$ 0.00	$ 0.00
TOTAL			$ 516452.13	$ 73571.92
PAVEMENT				
INTEREST RATE :	7.00 %			
SERVICE LIFE :	5 yrs.			
		123-456-7897	$ 2642532.50	$ 644488.75
		567-678-6789	$ 1887523.25	$ 460349.13
		756-143-6472	$ 3020037.25	$ 736558.63
TOTAL			$ 7550093.00	$ 1841396.50
SIDEWALK				
INTEREST RATE :	5.00 %			
SERVICE LIFE :	7 yrs.			
		234-345-4561	$ 429303.72	$ 74192.27
		567-678-6789	$ 192875.59	$ 33332.76
TOTAL			$ 622179.31	$ 107525.03
BARRIER CURB & GUTTER				
INTEREST RATE :	8.00 %			

SERVICE LIFE : 4 yrs.

		132-354-4657	$	42356.77	$ 12788.38
TOTAL			$	42356.77	$ 12788.38

ROLLED CURB & GUTTER
INTEREST RATE : 6.00 %
SERVICE LIFE : 6 yrs.

		132-354-4657	$	361478.50	$ 73511.27
		756-143-6472	$	361478.50	$ 73511.27
TOTAL			$	722957.00	$ 147022.55

BLVD. CURB & GUTTER
INTEREST RATE : 3.00 %
SERVICE LIFE : 8 yrs.

		234-345-4561	$	1714.93	$ 244.30
		567-678-6789	$	3429.86	$ 488.61
		756-143-6472	$	12004.51	$ 1710.12
TOTAL			$	17149.31	$ 2443.03

PARK DEVELOPMENT
INTEREST RATE : 7.00 %
SERVICE LIFE : 5 yrs.

		123-456-7897	$	4191.10	$ 1022.17
		567-678-6789	$	85.53	$ 20.86
TOTAL			$	4276.63	$ 1043.03

BUFFER DEVELOPMENT
INTEREST RATE : 5.00 %
SERVICE LIFE : 7 yrs.

		123-456-7897	$	1549.17	$ 267.73
		234-345-4561	$	2323.75	$ 401.59
		567-678-6789	$	3098.34	$ 535.45
		132-354-4657	$	387.29	$ 66.93
		756-143-6472	$	387.29	$ 66.93
TOTAL			$	7745.84	$ 1338.64

CATCH-BASIN
INTEREST RATE : 8.00 %
SERVICE LIFE : 4 yrs.

		123-456-7897	$	583.59	$ 176.20
		756-143-6472	$	5252.34	$ 1585.79
TOTAL			$	5835.94	$ 1761.99

MANHOLE
INTEREST RATE : 6.00 %
SERVICE LIFE : 6 yrs.

		234-345-4561	$	13253.38	$ 2695.24
TOTAL			$	13253.38	$ 2695.24

STREET LIGHT

INTEREST RATE : 3.00 %
SERVICE LIFE : 8 yrs.

	123-456-7897	$	5410.14	$	770.71
	234-345-4561	$	4508.45	$	642.26
	567-678-6789	$	3606.76	$	513.81
	756-143-6472	$	4508.45	$	642.26
TOTAL		$	18033.79	$	2569.03

STREET SIGN
INTEREST RATE : 7.00 %
SERVICE LIFE : 5 yrs.

	234-345-4561	$	2864.97	$	698.74
	567-678-6789	$	2864.97	$	698.74
TOTAL		$	5729.95	$	1397.48

TRAFFIC SIGN
INTEREST RATE : 5.00 %
SERVICE LIFE : 7 yrs.

	756-143-6472	$	9115.13	$	1575.28
TOTAL		$	9115.13	$	1575.28

SINGLE FAMILY
- GAS
INTEREST RATE : 8.00 %
SERVICE LIFE : 4 yrs.

	567-678-6789	$	3848.36	$	1161.90
	132-354-4657	$	7440.16	$	2246.34
	756-143-6472	$	14367.20	$	4337.75
TOTAL		$	25655.71	$	7745.99

- POWER
INTEREST RATE : 6.00 %
SERVICE LIFE : 6 yrs.

	234-345-4561	$	4598.31	$	935.12
	132-354-4657	$	41384.76	$	8416.12
TOTAL		$	45983.07	$	9351.24

- TELEPHONE
INTEREST RATE : 3.00 %
SERVICE LIFE : 8 yrs.

	123-456-7897	$	10895.37	$	1552.12
	567-678-6789	$	573.44	$	81.69
	132-354-4657	$	18350.10	$	2614.09
	756-143-6472	$	27525.15	$	3921.13
TOTAL		$	57344.05	$	8169.03

MULTI-FAMILY
- GAS
INTEREST RATE : 7.00 %
SERVICE LIFE : 5 yrs.

	567-678-6789	$	27686.52	$	6752.48

TOTAL				$	27686.52	$ 6752.48

- POWER

INTEREST RATE : 5.00 %

SERVICE LIFE : 7 yrs.

		123-456-7897	$	6933.55	$ 1198.26
		234-345-4561	$	1386.71	$ 239.65
		567-678-6789	$	9013.61	$ 1557.73
		132-354-4657	$	8666.93	$ 1497.82
		756-143-6472	$	8666.93	$ 1497.82
TOTAL			$	34667.73	$ 5991.28

- TELEPHONE

INTEREST RATE : 8.00 %

SERVICE LIFE : 4 yrs.

		756-143-6472	$	16000.86	$ 4830.99
TOTAL			$	16000.86	$ 4830.99

.0 OPERATION COST BREAKDOWN

OPERATION COST BREAKDOWN

DESCRIPTION	ACCOUNT NUMBER	PRESENT COST	ANNUAL COST
WATERMAIN			
INTEREST RATE : 6.34 %			
SERVICE LIFE : 9 yrs.			
	234-345-4561	$ 13943.22	$ 2080.40
	567-678-6789	$ 41829.67	$ 6241.21
TOTAL		$ 55772.89	$ 8321.61
DOMESTIC SEWER			
INTEREST RATE : 4.56 %			
SERVICE LIFE : 10 yrs.			
	123-456-7897	$ 41287.24	$ 5233.23
	567-678-6789	$ 39668.13	$ 5028.01
TOTAL		$ 80955.38	$ 10261.24
STORM SEWER			
INTEREST RATE : 3.99 %			
SERVICE LIFE : 7 yrs.			
	123-456-7897	$ 0.00	$ 0.00
	234-345-4561	$ 0.00	$ 0.00
	567-678-6789	$ 0.00	$ 0.00
	132-354-4657	$ 0.00	$ 0.00
TOTAL		$ 66820.99	$ 11128.92
PAVEMENT			
INTEREST RATE : 7.91 %			
SERVICE LIFE : 8 yrs.			
	123-456-7897	$ 306389.84	$ 53134.39
	567-678-6789	$ 218849.89	$ 37953.13
	756-143-6472	$ 350159.84	$ 60725.01
TOTAL		$ 875399.56	$ 151812.53
SIDEWALK			
INTEREST RATE : 5.67 %			
SERVICE LIFE : 11 yrs.			
	234-345-4561	$ 82143.15	$ 10240.11
	567-678-6789	$ 36904.89	$ 4600.63
TOTAL		$ 119048.05	$ 14840.74
BARRIER CURB & GUTTER			
INTEREST RATE : 6.34 %			

SERVICE LIFE : 9 yrs.

	132-354-4657	$	9549.92	$	1424.90
TOTAL		$	9549.92	$	1424.90

ROLLED CURB & GUTTER
INTEREST RATE : 4.56 %
SERVICE LIFE : 10 yrs.

	132-354-4657	$	78151.86	$	9905.89
	756-143-6472	$	78151.86	$	9905.89
TOTAL		$	156303.72	$	19811.78

BLVD. CURB & GUTTER
INTEREST RATE : 3.99 %
SERVICE LIFE : 7 yrs.

	234-345-4561	$	74.69	$	12.44
	567-678-6789	$	149.37	$	24.88
	756-143-6472	$	522.80	$	87.07
TOTAL		$	746.86	$	124.39

PARK DEVELOPMENT
INTEREST RATE : 7.91 %
SERVICE LIFE : 8 yrs.

	123-456-7897	$	51.53	$	8.94
	567-678-6789	$	1.05	$	0.18
TOTAL		$	52.58	$	9.12

BUFFER DEVELOPMENT
INTEREST RATE : 5.67 %
SERVICE LIFE : 11 yrs.

	123-456-7897	$	0.75	$	0.09
	234-345-4561	$	1.13	$	0.14
	567-678-6789	$	1.51	$	0.19
	132-354-4657	$	0.19	$	0.02
	756-143-6472	$	0.19	$	0.02
TOTAL		$	3.77	$	0.47

CATCH-BASIN
INTEREST RATE : 6.34 %
SERVICE LIFE : 9 yrs.

	123-456-7897	$	93.66	$	13.97
	756-143-6472	$	842.94	$	125.77
TOTAL		$	936.60	$	139.75

MANHOLE
INTEREST RATE : 4.56 %
SERVICE LIFE : 10 yrs.

	234-345-4561	$	1960.21	$	248.46
TOTAL		$	1960.21	$	248.46

STREET LIGHT

INTEREST RATE : 3.99 %
SERVICE LIFE : 7 yrs.

	123-456-7897	$	259.17	$	43.16
	234-345-4561	$	215.98	$	35.97
	567-678-6789	$	172.78	$	28.78
	756-143-6472	$	215.98	$	35.97
TOTAL		$	863.91	$	143.88

STREET SIGN
INTEREST RATE : 7.91 %
SERVICE LIFE : 8 yrs.

	234-345-4561	$	110.59	$	19.18
	567-678-6789	$	110.59	$	19.18
TOTAL		$	221.17	$	38.36

TRAFFIC SIGN
INTEREST RATE : 5.67 %
SERVICE LIFE : 11 yrs.

	756-143-6472	$	269.06	$	33.54
TOTAL		$	269.06	$	33.54

SINGLE FAMILY
- GAS
INTEREST RATE : 6.34 %
SERVICE LIFE : 9 yrs.

	567-678-6789	$	841.65	$	125.58
	132-354-4657	$	1627.20	$	242.79
	756-143-6472	$	3142.18	$	468.83
TOTAL		$	5611.03	$	837.20

- POWER
INTEREST RATE : 4.56 %
SERVICE LIFE : 10 yrs.

	234-345-4561	$	907.81	$	115.07
	132-354-4657	$	8170.32	$	1035.60
TOTAL		$	9078.13	$	1150.67

- TELEPHONE
INTEREST RATE : 3.99 %
SERVICE LIFE : 7 yrs.

	123-456-7897	$	1152.53	$	191.95
	567-678-6789	$	60.66	$	10.10
	132-354-4657	$	1941.10	$	323.29
	756-143-6472	$	2911.65	$	484.93
TOTAL		$	6065.95	$	1010.27

MULTI-FAMILY
- GAS
INTEREST RATE : 7.91 %
SERVICE LIFE : 8 yrs.

	567-678-6789	$	2768.23	$	480.07

TOTAL				$	2768.23	$ 480.07

- POWER

INTEREST RATE : 5.67 %
SERVICE LIFE : 11 yrs.

		123-456-7897	$	1043.96	$ 130.14
		234-345-4561	$	208.79	$ 26.03
		567-678-6789	$	1357.14	$ 169.18
		132-354-4657	$	1304.95	$ 162.68
		756-143-6472	$	1304.95	$ 162.68
TOTAL			$	5219.79	$ 650.71

- TELEPHONE

INTEREST RATE : 6.34 %
SERVICE LIFE : 9 yrs.

		756-143-6472	$	3333.97	$ 497.45
TOTAL			$	3333.97	$ 497.45

D.31 SUMMARY OF ACCOUNTS

SU .Y OF ACCOUNTS :

DESCRIPTION	ACCOUNT NUMBER	CONSTRUCTION COST	MAINTENANCE COST	OPERATION COST
Aloysius & Sons Rotorooters	123-456-7897	$ 2636778.00	$ 685450.19	$ 40572.75
Sammy's Cement Company	234-345-4561	$ 1093292.75	$ 94535.23	$ 7446.04
City of Cucamonga	567-678-6789	$ 2659762.00	$ 592433.63	$ 36855.85
Clancy's Construction	132-354-4657	$ 107251.54	$ 98282.79	$ 7616.33
Bonafide Buildings Limited	756-143-6472	$ 2198006.25	$ 823969.25	$ 49548.46
T O T A L		$ 12264964.00	$ 2388494.25	$ 222966.05