ENGINEERS' GUIDE
to
ICES COGO I

First Edition
August, 1967

R67-46

MIT

DEPARTMENT
OF
CIVIL
ENGINEERING

SCHOOL OF ENGINEERING
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
Cambridge, Massachusetts 02139

ENGINEERS' GUIDE

to

ICES COGO I

First Edition

August, 1967

Civil Engineering Systems Laboratory

Massachusetts Institute of Technology

Cambridge, Massachusetts

FOREWORD

The advances reported herein were developed at the M.I.T. Civil Engineering Systems Laboratory (CESL), a research unit of the Department of Civil Engineering dedicated to the advancement of the teaching and practice of civil engineering.

This document is one of a series of publications resulting from the Integrated Civil Engineering System (ICES) Project, initiated and carried out under the general direction of Professor C. L. Miller, Head of the Department. The ICES Project is a cooperative venture of government, industry, and university groups interested in the development of a large-scale, computer-based system which integrates advanced information system and powerful problem-solving capabilities. An operational system has been initially implemented on the IBM System/360.

Support for the ICES Project has been provided by a number of CESL research sponsors. Special recognition is due the following: the IBM Corporation, the Massachusetts Department of Public Works in cooperation with the U.S. Bureau of Public Roads, the Massachusetts Bay Transportation Authority, the National Science Foundation, the McDonnell Automation Company, the Wisconsin State Highway Commission, the Ford Foundation, and the Union Pacific Railroad Foundation.

Additional information on the availability of the publications and computer systems resulting from the ICES Project may be obtained from Professor Daniel Roos, Director, Civil Engineering Systems Laboratory, Room 1-163, Massachusetts Institute of Technology, Cambridge, Massachusetts, 02139.

CONTRIBUTORS

A number of people have participated in the development of ICES COGO I at the M.I.T. Civil Engineering Systems Laboratory during the period 1964 to 1967. The principal contributors and their contributions include:

C. L. Miller

Formulated the over-all system and designed the language; developed the external specifications and wrote the Engineers' Guide; provided over-all supervision.

Steven B. Lipner

Supervised the implementation of the system, including supervision of and responsibility for all programming development and system documentation; programmed and optimized major portions of the system.

Betsy Schumacker

Programmed the complete Traverse subsystem and developed the file save and restore capability; served as systems consultant to the systems design and development staff.

Daniel Roos

Designed the internal data structure; formulated the internal design and development of COGO in the ICES environment.

T. H. Kaalstad and E. E. Newman

Performed testing and evaluation of completed commands; assisted in technical development, administration, and production of completed system.

* * * * * * * * * * *

Many CESL staff members have contributed to the programming of the system. Special recognition is due to the following:

Thomas A. Casey
Kerry B. Chesbro
Larry-Stuart Deutsch
Michael R. Dunlavey
Kenneth G. Follansbee
Lawrence A. Hall
Carl Jones, III
Marc R. Levin
Paul M. Martin
John C. Prokopy
Henry A. Radzikowski, II
Angel (7) Silva
Howard C. Stotland

ENGINEERS' GUIDE

to

ICES COGO I

* * * * * * * * * * * * *

PART 1

GENERAL SPECIFICATIONS

Civil Engineering Systems Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

TABLE OF CONTENTS

INTRODUCTION

ICES COGO I is an information processor for the computer solution of geometric problems in such areas as

Engineering surveying and mapping
Highway, railway, waterway, and airport design
Structural, bridge, and building design
Right-of-way and land acquisition
Construction layout and control
Subdivision design and layout
Utility surveying, mapping, and records
Cadastral surveys and property records
Urban planning and design geometrics
Military and aerospace installation design
Architectural design geometrics and site planning

The processor is made up of a language, a set of processing routines, and a set of information files. While the COGO (for COordinate GeOmetry) language is designed for the natural expression of geometric problems in the above areas, the system may be applied to a variety of geometric problems in any area which involves points, lines, curves, and polygons in two or three dimensional space.

ICES COGO I operates on geometric objects and variables which can be identified, stored, retrieved, computed, printed, and manipulated by means of commands issued by the Engineer-User. The purpose of this document is to describe the external specifications of the processor as a guide to the Engineer-User. Part 1 describes the general specifications for the processor, and Part 2 describes the detailed specifications for the individual commands.

PROBLEM-SOLVING APPROACH

Information Processor

The COGO information processor operates in the ICES environment and includes the following components:

a) a command-structured, problem-oriented language which the Engineer-user utilizes to compose commands which describe the solution procedures, results, and data for problems in the area of civil engineering geometrics;

b) a set of processing routines which are executed by the computer to achieve the problem-solving actions specified by the commands;

c) a set of information files or data tables which provide for the storage and retrieval of geometric objects which are components of the problems being solved.

The language provides the mechanism for the Engineer to communicate with the problem-solving capabilities of the processing routines and with the information files. Through commands the Engineer describes:

a) the known data for the variables and problem components;

b) the relationships between the variables and problem components;

c) the information storage, retrieval, and processing to be performed;

d) the results to be recorded.

The language and the commands are intended to provide a natural, flexible, and general way of providing such descriptions for geometric problems.

The language is machine independent; that is, it is independent of both the hardware and software of the computer. The Engineer need not be aware of the computer per se to use the language. He does not need to be proficient in computer programming at lower language levels nor knowledgeable of computer hardware. The language isolates him from the need for such understanding. Any such understanding he does have will give him insight as to the functions of the computer but is not required for use of the subsystem.

The processing routines are machine dependent; that is, they must exist as machine executable code for a particular computer system. The processing routines are the province of the computer facility to which the Engineer submits his COGO problems for processing. The processing routines take the form of software--an organized set of programs, subprograms, and subroutines in machine language which are executed by the hardware. Given the existence of the processing routines at an available facility, the Engineer need only concern himself with the language and local procedures for submitting COGO problems to the facility.

The information files are automatically generated by the processing routines when commands which define new geometric objects are executed. The information files are contained in the physical storage devices of the computer--the core memory and, when needed, the secondary storage devices such as disks. The Engineer can save his information files between runs if he so specifies.

In order to use COGO, the Engineer must of course understand civil engineering geometric problem-solving and his particular problem. There is nothing automatic about the system with respect to problem formulation and solution description. These are entirely the province of the Engineer. Like writing a speech, knowing the language is not enough. One must assemble the words in meaningful and effective ways to convey a message. With COGO, the Engineer is fully responsible for assembling the commands in such a way that he achieves the desired solution to the problem which he formulates. Writing COGO is an art, proficiency in which improves with practice, but again like speech writing, requires that one know his subject.

Geometric Objects and Data Tables

Individual problems in civil engineering geometrics can be expressed as a particular configuration of simple geometric objects and a particular combination of known and unknown values. Examples of objects include points, lines, courses, curves, alignments, parcels, profiles, and so forth. Examples of values include distances, angles, directions, coordinates, stations, and so forth. The objective of the COGO language is to allow the Engineer to specify the particular combination and configuration which describe and solve his particular problem rather than being restricted in any way by a predetermined or programmed combination and configuration.

The geometric objects comprise the building blocks for formulating and solving a geometric problem. Individual objects are identified, defined, and stored in appropriate data tables via commands. Once stored, an object can be referred to in subsequent commands and used to define other objects. Examples of identified objects which might appear in commands include:

POINT 24	COURSE 'B14'	ALIGNMENT 'RAMP-K'
LINE 15	PARCEL 'P23-A8'	BASELINE 'B5'
CURVE 78	TRAVERSE 'T4'	PROFILE 'RC/L'

Scalars or single value variables may also be identified, defined, stored, and referred to in commands, such as

DISTANCE 'X2' AZIMUTH 'A8' ANGLE 'JACK'

The data tables store the information necessary to construct the object so that it can be used in computations and information processing.

Commands

A command is a prescribed set of words, objects, and data items which defines a new object or specifies some action requested by the Engineer. Examples of commands include:

```
STORE POINT 17 N 1000. E 2000. STA 5+00 Z 150.15
LOCATE POINT 2 FROM POINT 4, DISTANCE 125.16, AZIMUTH
    134 15 25
LOCATE POINT 8, INTERSECT CURVE 24 WITH LINE 15
PRINT ANGLE AT POINT 4 FROM POINT 2 TO POINT 4792
STORE ALIGNMENT 'K', 2, 7, CURVE 15, CURVE 38, COURSE 'A', 24
DESCRIBE PARCEL 'LOT/18'
```

The first word (such as STORE, LOCATE, PRINT) is the command name. Subsequent words (such as POINT, DISTANCE, INTERSECT) are labels for data items or objects or are command modifiers which specify the particular combination of objects and data items and the processing action to be taken.

The design of the language and the allowable forms for the commands are intended to provide the Engineer with a natural way to express the formulation and solution of his problem as a series of readable technical phrases. However, when conciseness is preferred, abbreviated forms of the commands may be used. Command words and modifiers may be abbreviated. Optional words which allow for readability may be omitted. If the data items and objects are in the standard form and order, the labels may often be omitted. The following are two extremes of the same command:

```
LOCATE POINT 4 FROM POINT 8, DISTANCE FROM POINT 7 TO
    POINT 3, AZIMUTH 'A5' PLUS 90
LOC 4, 8, 7 TO 3, 'A5' P 90
```

Accordingly, the form of a command can vary from a complete readable technical phrase to a single abbreviated word followed by a string of data values, or any combination in between.

Command Processing and Execution

The commands written by the Engineer serve as the input to the computer and the COGO processing routines. In an off-line or batch mode of operation, the Engineer writes his commands out on paper and then has them punched onto cards. The deck of cards is then read into the computer via a local or remote card reader as the input device. Output is recorded via a local or remote printer as the output device. In an on-line mode of operation the Engineer may input his commands directly to the computer, working with a typewriter type unit as the input and output device. The on-line mode has significant advantages, particularly in an interactive computer environment when the Engineer can inspect the results of each command before inputting the next.

The commands are processed individually, one at a time, and in the order in which they are received by the computer. Each command may be thought of as a small program which

a) is processed and executed independently of the commands which precede and follow;

b) may make use of data stored in the Data Tables by the commands which precede it; and

c) may store data in the Data Tables for use by the commands which follow.

Accordingly, COGO uses an incremental problem-solving approach, each command adding an increment to the formulation and solution of the problem. Small, simple problems require a small number of commands to build up a solution. Large, complex problems require a large number of commands. Since there is no practical restriction on the number of commands which may be used in solving a problem, there is no practical restriction on the size or complexity of problems which can be handled. A problem may involve as few as 3 or 4 commands or as many as several thousand or more commands. The Engineer may formulate his problem in as large or small terms as are most convenient and natural. What constitutes a problem and the commands to build up the solution are under his full control.

The processing and execution of each command is the work of the processing routines. Consider the following command:

LOCATE POINT 4, INTERSECT CURVE 5 WITH LINE 6, PRINT

The action of the processing routines is roughly as follows:

1) An executive routine determines that the command is a Locate command and stores the information from the command.

2) The data for Curve 5 is retrieved from the Curve Data Table, and the data for Line 6 is retrieved from the Line Table.

3) The subroutine for intersecting a curve with a line is retrieved and executed to compute the coordinates of Point 4.

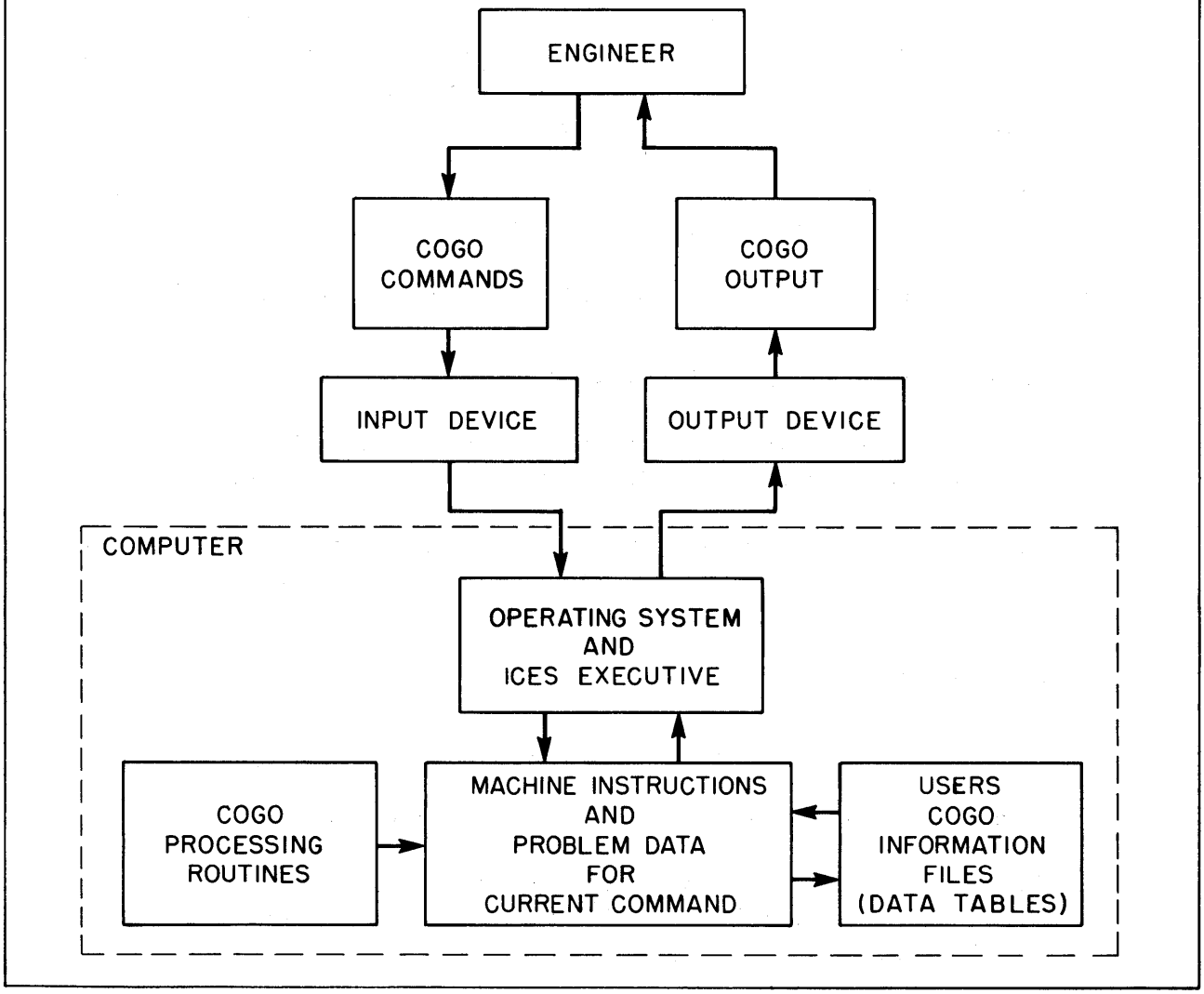
4) The coordinates of Point 4 are stored in the Point Table.

5) The coordinates of Point 4 are printed in the output report.

6) Control returns to the executive to read and process the next command (which may refer to Point 4 since it is now stored in the Point Table).

If for any reason the command cannot be processed or executed (such as command improperly constructed, Curve 5 or Line 6 not stored, no intersection, and so forth), appropriate messages are printed and corrective action, if appropriate, taken. An extensive part of the work of the processor is concerned with checking the validity of the command and in generating diagnostics to guide the Engineer when trouble is encountered.

SIMPLIFIED DIAGRAM OF COGO COMMAND PROCESSING



Example Problem

The problem-solving approach may be illustrated by a very simple example problem. The problem sketch, Engineer's solution, listing of the input commands, and listing of the output for the example problem are given in the plates which follow. The explanation of commands in the Engineer's solution is as follows:

1) The COGO Information Processor is specified.

2) The System is set for particular input/output conventions.

3) A comment is given.

4) The known coordinates of Point 4 are stored in the Point Table.

5) The coordinates of Point 8 are computed, using a numerical distance and direction, and are stored in the Point Table.

6) Curve 3 is defined and stored in the Curve Table. One of many possible ways of defining and storing a curve is used.

7) The distance from the PI of Curve 3 to Point 2 is assigned name 'X1', computed as half the distance between Point 4 and PI of Curve 3, and is stored in the Distance Table. The value of the distance is to be printed.

8) The coordinates of Point 2 are computed, using stored distance 'X1' and a computed direction between two stored points, and are stored in the Point Table.

9) The coordinates of Point 7 are computed, by intersecting a line defined by a point and a computed direction with a course defined by two stored points, and are stored in the Point Table.

10) Parcel 'LOT/18' is defined by a list of stored objects and is stored in the Chain Table.

11) Values for distances between stored points are requested.

12) The value of an angle defined by stored points is requested.

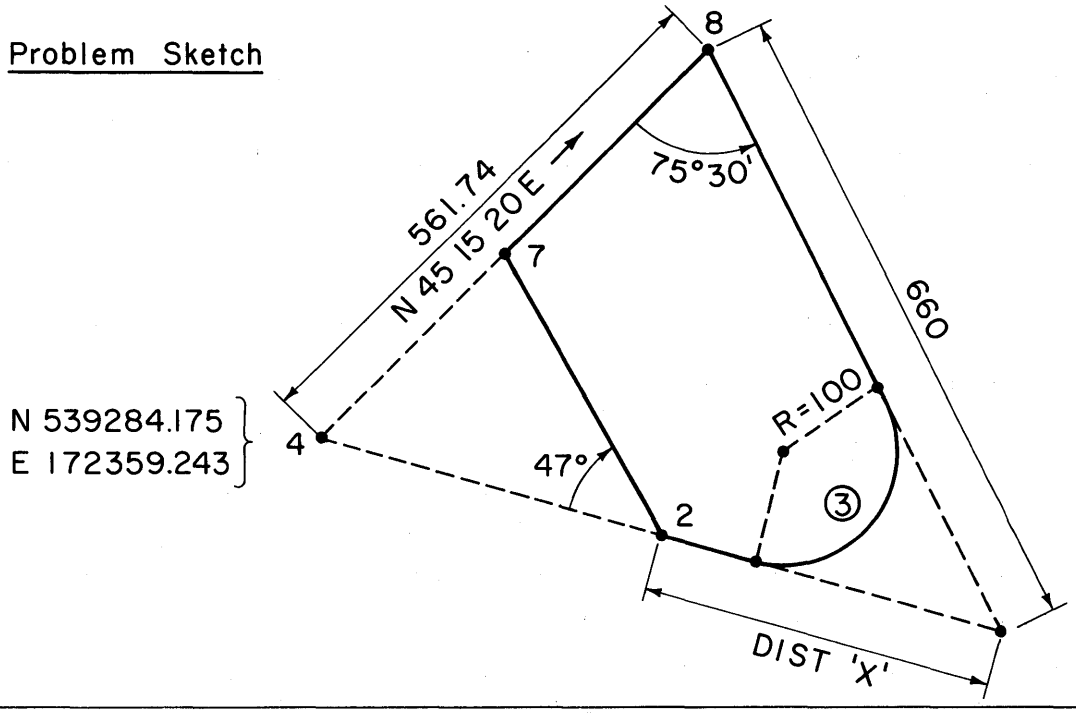
13) A complete report describing all values associated with the stored objects which define 'LOT/18' is requested. The coordinates of points and curve points, curve elements, lengths and directions of all sides, area, and so forth, are to be printed.

14) The end of the computer run is specified.

While a very small subset of the total capabilities is illustrated in this simple problem, the concept of incremental problem-solving and the building up of a solution with a set of flexible commands is illustrated.

EXAMPLE PROBLEM

Problem Sketch



Engineer's Solution (Long and Short Form)

```
(1) COGO
(2) SET SYSTEM NE, BEARINGS
(3) $ LOT/18 IN SUNSHINE SUBDIVISION
(4) STORE POINT 4 N 539284.175 E 172359.243
(5) LOCATE POINT 8 FROM POINT 4 DISTANCE 561.74 BEARING N45 15 20E
(6) STORE CURVE 3, PB AT 8, DB 8 TO 4 M 75 30, TTL 660. R 100. PA AT 4
(7) STORE DISTANCE 'X1', PI 3 TO 4 DIVIDED BY 2.0, PRINT
(8) LOCATE 2 FROM PI 3, DIST 'X1' AZ PI 3 TO 4
(9) LOCATE 7 INTERSECT COURSE 4 TO 8 WITH LINE THRU 2 AT AZ 2 TO 4 P 47
(10) STORE PARCEL 'LOT/18' 7, 8, CURVE 3, 2, 7
(11) PRINT DISTANCE 4 TO 7, 4 TO 2
(12) PRINT ANGLE AT 4 FROM 8 TO PT 3
(13) DESCRIBE PARCEL 'LOT/18'
(14) FINISH
```

```
(1) COGO
(2) SET NE, BEA
(3) $ LOT/18 IN SUNSHINE SUBDIVISION
(4) S 4, 539284.175 172359.243
(5) LOC 8 4, 561.74 N 45 15 20 E
(6) S CUR 3, PB 8, DB 8 TO 4 M 75.5, TTL 660, R 100, PA4
(7) D DIS 'X1' PI 3 TO 4 DIV 2, PRI
(8) LOC 2, PI 3, 'X1' PI 3 TO 4
(9) D 7 INT LINE THRU 2, 2 TO 4 P 47, WITH COU 4 TO 8
(10) S PAR 'LOT/18' 7, 8, CUR 3, 2, 7
(11) P DIS 4 TO 7, 4 TO 2
(12) D ANG AT 4, 8, PT 3
(13) DES PAR 'LOT/18'
(14) FINISH
```

Listing of Output

```
COGO
WELCOME TO COGO,YOUR FRIENDLY COORDINATE GEOMETRY SYSTEM

SET SYSTEM NE,BEARINGS

$ LOT/18 IN SUNSHINE SUBDIVISION

STORE POINT 4  N 539284.175  E 172359.243

LOCATE POINT 8 FROM POINT 4  DISTANCE 561.74  BEARING N 45 15 20 E

STORE CURVE 3, PB AT 8,DB 8 TO 4 M 75 30, TTL 660., R 100. PA AT 4

STORE DISTANCE 'X1',PI 3 TO 4 DIVIDED BY 2.0, PRINT
    DISTANCE X1       375.997

LOCATE 2 FROM PI 3, DIST 'X1'  AZ PI 3 TO 4

LOCATE 7 INTERSECT COURSE 4 TO 8 WITH LINE THRU 2 AT AZ 2 TO 4 P 47

STORE PARCEL 'LOT/18' 7,8,CURVE 3,2,7

PRINT DISTANCE 4 TO 7, 4 TO 2
    DISTANCE FROM      4 TO      7     294.929
    DISTANCE FROM      4 TO      2     375.997

PRINT ANGLE AT 4 FROM 8 TO PT 3
    ANGLE AT      4 FROM      8 TO PT   3     58 10 48.58

DESCRIBE PARCEL 'LOT/18'



   DESCRIPTION OF CHAIN LOT/18

               CHAIN ELEMENTS

    COURSE      FROM      7 TO      8        LENGTH   276.811        BEARING  N 45 15 20.00 E
    COURSE      FROM      8 TO PC   3        LENGTH   426.231        BEARING  S 30 14 40.00 E

    CURVE   3    TYPE C       CURVE ELEMENTS

         RADIUS          100.000  DEGREE  57 17 44.81
         LENGTH          233.316  DELTA  133 40 48.58
         TANGENT         233.769  BACK  S 30 14 40.00 E
         EXTERNAL        154.260  AHEAD N 76 33 51.42 W
         LONG CHORD      183.882
         MID. ORD.        60.670


    COURSE      FROM PT   3 TO      2        LENGTH   142.228        BEARING  N 76 33 51.42 W
    COURSE      FROM      2 TO      7        LENGTH   331.040        BEARING  N 29 33 51.42 W



               CHAIN POINTS

         POINT       7       N  539484.750     E  172561.615     S ************     Z ************
         POINT       8       N  539679.610     E  172758.221     S ************     Z ************

    CURVE   3    TYPE C       CURVE POINTS

         POINT CC    3       N  539261.026     E  172886.521     S ************     Z ************
         POINT PC    3       N  539311.395     E  172972.909     S      0+ 0.0      Z ************
         POINT PT    3       N  539163.763     E  172863.285     S      2+33.316    Z ************
         POINT PI    3       N  539109.446     E  173090.656     S      2+33.769    Z ************


         POINT       2       N  539196.810     E  172724.950     S ************     Z ************
         POINT       7       N  539484.750     E  172561.615     S ************     Z ************

    AREA SEGMENT FROM PC TO PT ON CURVE   3        8049.763 SQUARE FEET


    TOTAL AREA OF LIST       122242.692 SQUARE FEET      2.806 ACRES

FINISH
        THE ABOVE COGO PROBLEM INCLUDED    0 ERRORS WHICH REQUIRED COMMAND ABORT.
        OF THESE COMMANDS    0 INVOLVED ATTEMPTS TO RETRIEVE UNSTORED OBJECTS.
```

DEFINITION OF OBJECTS

Object Definition

Each object to be stored in a data table is given a unique name or identification number in the command which stores the object. When the command is executed, entries which provide the values or information to define the object are made in the data table. The entries stored in the data table remain unchanged unless the object is redefined in a subsequent command. An object is redefined if the same name or number is reused to store an object. The new table entries then replace the old entries. When an object is redefined, a message is printed in the output to alert the Engineer in case the redefinition was unintentional.

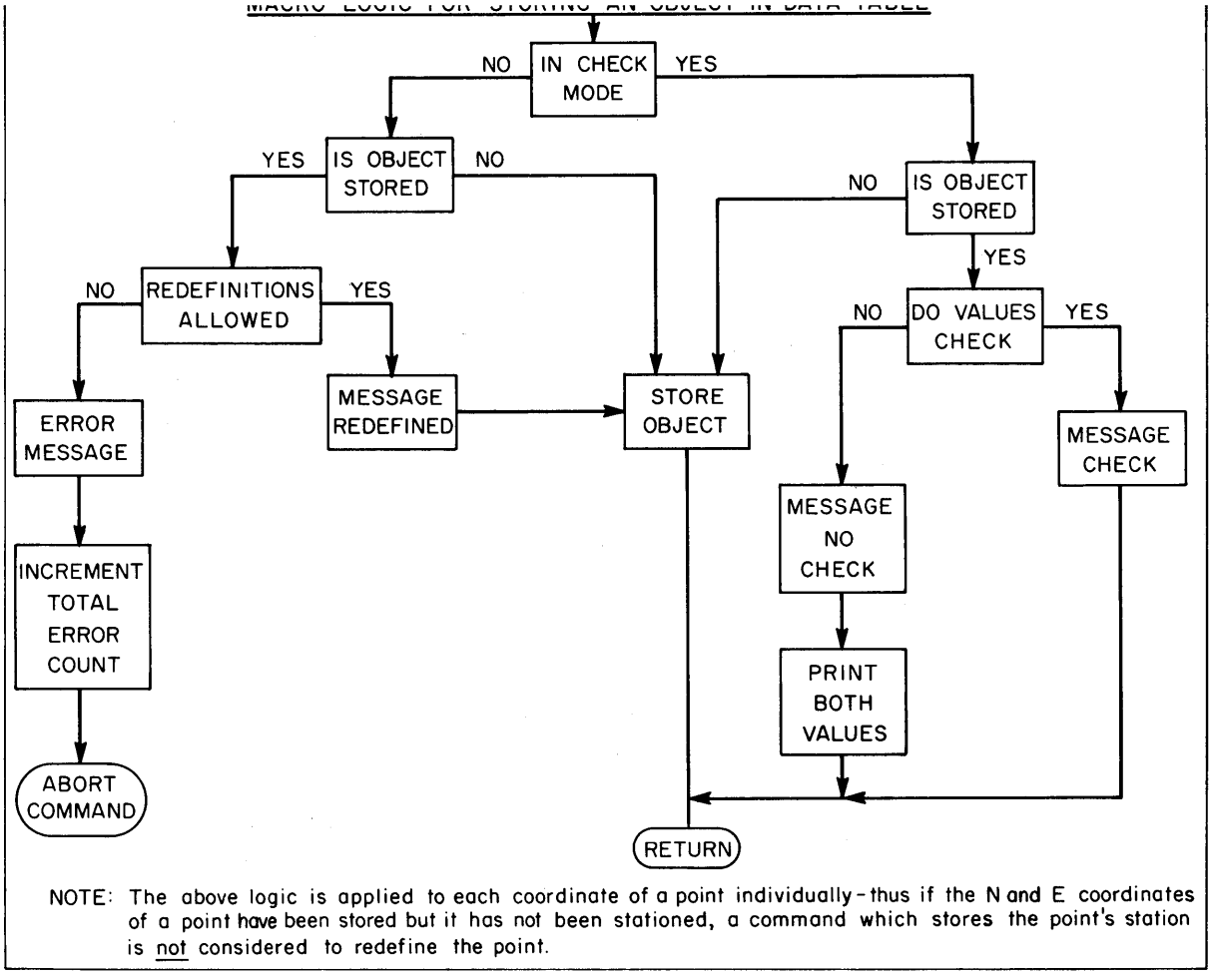
Once an object is defined (stored in the data table), it may be referred to or used as a data item in subsequent commands. Use of an object as a data item does not affect the table entries. If the Engineer attempts to compute with an undefined object, the command containing the undefined object will not be executed, an appropriate error message being printed. The system keeps count of the number of commands with one or more undefined objects. When the number of such commands exceeds a set limit, the execution of commands is terminated, the remaining commands being edited but not processed.

Absolute and Relative Objects

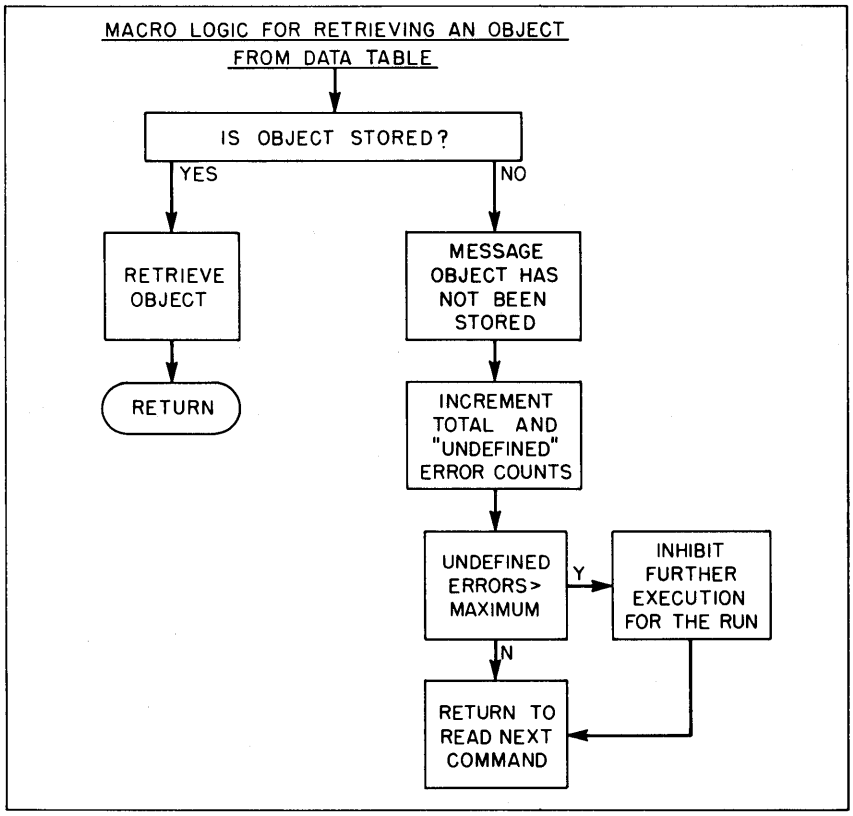
An absolute object is one with table entries which are numerical data values. An absolute object is independent of other objects in other data tables and is independent of the objects used as data items to define it when it was stored. A point is an example of an absolute object, with numerical values for coordinates, station, and elevation as the table entries. A relative object is one stored in terms of other objects in other tables. A relative object is dependent on the stored objects which define it. If they are redefined, the relative object is redefined. A course is an example of a relative object, being defined in the Course Table by the identification numbers of two points which are in turn stored in the Point or Curve Tables.

Deletion of Stored Objects

Stored objects may be deleted from the data tables if desired by use of the Delete command. Since the capacity of the system is quite large compared to usual problem sizes, the need to delete objects to free storage should be quite rare. However, some Engineers may prefer to reuse blocks of identification numbers and names of convenient (small) size in the same problem. The Delete capability will be useful in such instances.



NOTE: The above logic is applied to each coordinate of a point individually - thus if the N and E coordinates of a point have been stored but it has not been stationed, a command which stores the point's station is not considered to redefine the point.



DATA TABLES

The following data tables are maintained by the system:

Point Table

Each point to be stored is assigned an identification number which can be any integer from 0 to 9999. The stored data for each point is the horizontal coordinates, station, elevation or any subset of these items. A point is an absolute object. The Point Table has a capacity of 10,000 points.

Line Table

Each line to be stored is assigned an identification number which can be any integer from 0 to 999. A line is defined by a stored point and a direction. The stored data for each line is the point number and the direction magnitude. The direction is absolute, but a line is a relative object which moves with the point which defines it. The Line Table has a capacity of 1,000 lines.

Course Table

Each course to be stored is assigned a name which can be any 1 to 4 characters enclosed in single quotes. A course is defined as the line segment between two stored points. The stored data for each course is the point numbers for the beginning and end points. A course is a relative object which moves with the points which define it.

Curve Table

Each horizontal curve to be stored is assigned an identification number which can be any integer from 0 to 999. A curve is defined as a circular curve segment with or without equal or unequal transition spirals. The stored data for each curve includes the horizontal coordinates and station of each curve point (PI, PC, PT, TS, ST, and so forth) and the elements of the curve (radius, length, and so forth). A curve is an absolute object. The Curve Table has a capacity of 1,000 curves.

Chain Table

Horizontal alignments, traverses, parcels, ramps, right-of-way lines, baselines, and similar kinds of horizontal geometry objects are stored in the Chain Table. Such objects are defined and stored as lists of other stored objects, treated as "links" which when connected form the continuous object or "chain." The list may include any combination of stored points, courses, curves, and other chains. Each chain to be stored is assigned a name which can be any 1 to 8 characters enclosed in single quotes. The stored data for each chain is the list of objects which define it. A chain is a relative object which moves with the objects which define it.

Profile Table

Vertical alignments are stored in the Profile Table. Profiles are defined by the stored points at the VPI's and the symmetric or asymmetric curve lengths. Each profile to be stored is assigned a name which can be any 1 to 8 characters enclosed in single quotes. The stored data for each profile is a table of point numbers and vertical curve lengths. The profile is a relative object which moves with the points which define it.

Scalar Tables

Separate tables are maintained for distances, angles, and directions--a Distance Table, Angle Table, and Direction Table. Each scalar to be stored is given a name which can be any 1 to 4 characters enclosed in single quotes. The stored data for each scalar is its absolute numerical value.

Text Table

Descriptive information associated with a stored chain may be stored in the Text Table. Each set of text to be stored is assigned the same name as that of the chain with which it is associated. The text is printed whenever the chain is described in an output report. The text feature allows non-geometric information, such as ownership, value, buildings, land use, and so forth, to be stored with objects such as parcels.

Objects stored in different tables may have the same name or number. For example, POINT 4, LINE 4, CURVE 4 each refer to a different object. DISTANCE 'A', AZIMUTH 'A', and ANGLE 'A' refer to different scalars.

SUMMARY OF DATA TABLES

Object Label	Id. Number or Name**	Table Capacity	Table Entries	Absolute or Relative
Point POINT	Numeric 1 to 4 digits 0 to 9999	10,000 Points	Horizontal Coord. Station Elevation	Absolute
Distance DISTANCE	Alphanumeric 1 to 4 characters	Open*	magnitude (Absolute value)	Absolute
Angle ANGLE	Alphanumeric 1 to 4 characters	Open*	magnitude (Absolute value)	Absolute
Direction AZIMUTH BEARING	Alphanumeric 1 to 4 characters	Open*	magnitude (Absolute value)	Absolute
Line LINE	Numeric 1 to 3 digits 0 to 999	1,000 Lines	Point Number Direction	Relative Absolute
Course COURSE	Alphanumeric 1 to 4 characters	Open*	Beginning and End Point Numbers	Relative
Curve CURVE	Numeric 1 to 3 digits 0 to 999	1,000 Curves	Coordinates and Station of all Curve Points. Curve Elements	Absolute
Chain CHAIN TRAVERSE ALIGNMENT PARCEL	Alphanumeric 1 to 8 characters	Open*	Chain List	Relative
Profile PROFILE	Alphanumeric 1 to 8 characters	Open*	VPI Points VC lengths	Relative Absolute

* Capacity a function of hardware configuration but not a practical constraint.

** Alphanumeric names must be enclosed in single quotes.

OBJECT GEOMETRY AND NOMENCLATURE

Point

A point is defined geometrically by its horizontal coordinates, station, and elevation or any subset thereof, such as horizontal coordinates only, station and elevation only, horizontal coordinates and station, and so forth. Points stored in the Point Table are termed integer points when it is necessary to distinguish them from curve points.

Externally, the coordinate system being used may be X, Y or N, E (Y followed by X), as determined by the system setting. Internally, the system is X, Y. A point is stored internally as the numerical values for the individual data items which are defined (coordinates, elevation, station). If entries are made in the Point Table for a point already stored, the new values for the individual data items replace the old values, the old values being retained for those data items unchanged. Thus selective entries can be made in the Point Table, as illustrated in the following examples:

Commands	Resulting Table Entries			
	X	Y	Z	S
STORE POINT 2 X 100. Y 200.	100.	200.	****	****
STORE POINT 2 X 5+00.	100.	200.	****	500.
STORE POINT 2 X 600. Y 300. Z 400.	600.	300.	400.	500.

X and Y are defined after the first command, and S is also defined after the second. All data values are defined after the third command. The third command redefines the point because defined table entries are changed.

When stored points are printed, the numerical values are printed for the defined data items and a row of asterisks for those undefined.

Curve Points

Curve points of stored curves may be used as data items in many commands. The convention for referring to a curve point is to give the curve point label followed by the stored curve member. For example:

PC 4	means the PC of Curve 4
PI 8	means the PI of Curve 8
SC 2	means the SC of Curve 2

The distance data item

DISTANCE PT 3 TO TS 5

means the distance from the PT of Curve 3 to the TS of Curve 5.

Directions

Externally, directions may be given as bearings or azimuths, interchangably. The azimuths may be measured from North or from South, as determined by the system setting. Numerical directions and angles can be input as

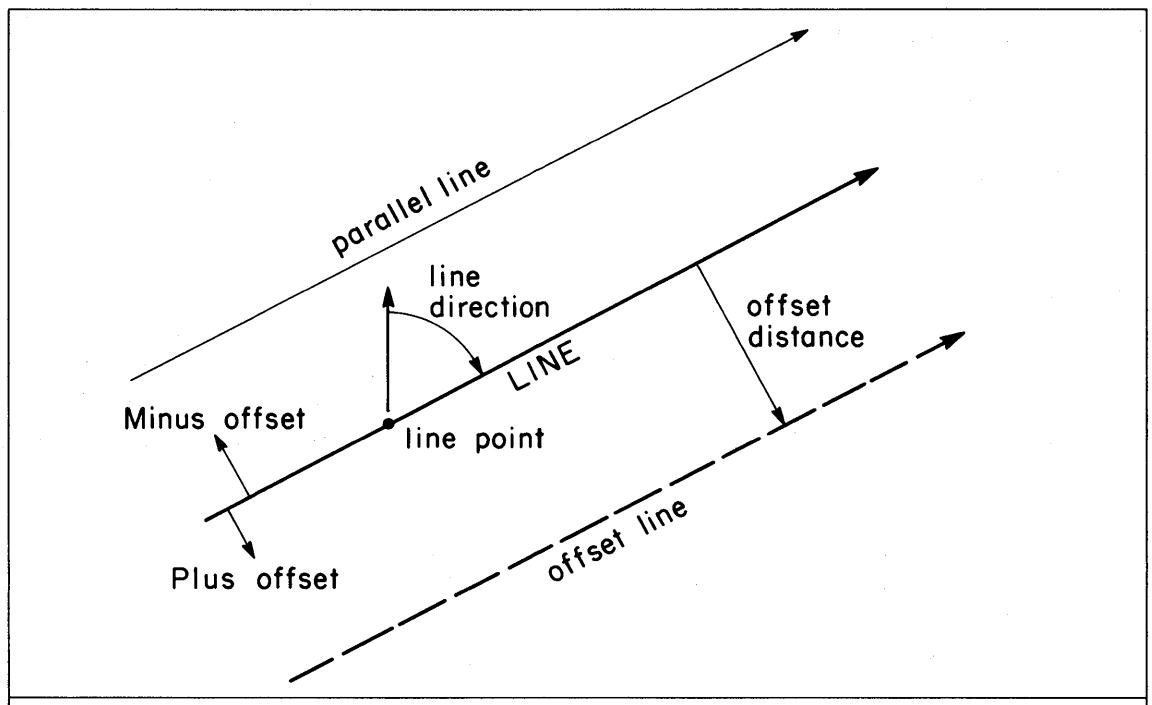
degrees, minutes, and decimal seconds
degrees, minutes, and even seconds
degrees and decimal minutes
degrees and even minutes
decimal degrees
even degrees

Internally, the system works with azimuths from north in decimal radians from 0 to $+2\pi$.

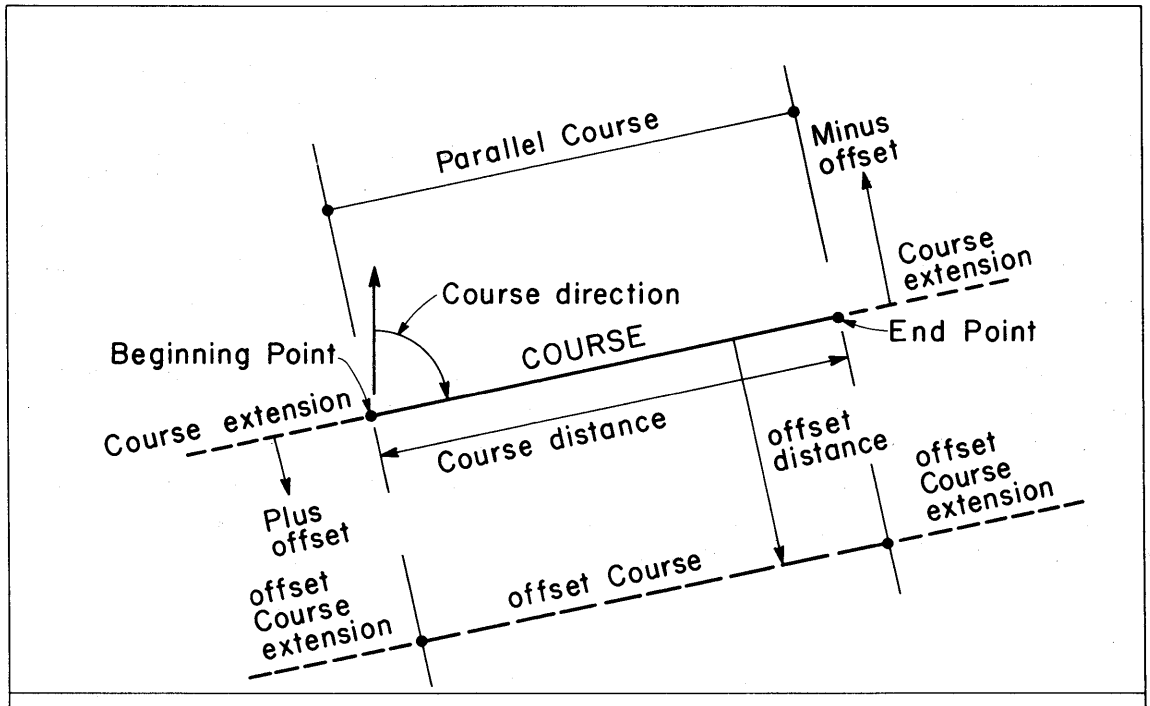
Line

A line is defined geometrically by the horizontal coordinates of a stored point on the line (the line point) and the direction of the line. A line has position in space and a forward direction but has infinite length. In computing with a line, the line is extended ahead and back from the line point.

A line is stored internally in the Line Table as the identification number of the line point (which may be a curve point) and the numerical value of the direction of the line. In computing with a stored line, the horizontal coordinates of the line point are automatically retrieved from the Point Table or Curve Table. Accordingly, the horizontal coordinates of the line point must be stored. If the line point is redefined (coordinates changed), the line is shifted (position changed) parallel to itself as the line direction is not changed. Hence a line is a relative object with respect to the line point but an absolute object with respect to its direction.



LINE OBJECT NOMENCLATURE



COURSE OBJECT NOMENCLATURE

Course

A course is defined geometrically as a line segment between two stored points. A course has length and direction, a beginning point and an end point. Its length is the distance between the two points, and its direction is the direction of the line from the beginning point toward the end point.

A course is stored internally in the Course Table as the identification numbers of the two points (which may be curve points) defining it. In computing with a stored course, the horizontal coordinates of the points which define it are automatically retrieved from the Point Table or Curve Table. Accordingly the horizontal coordinates of the points must be stored. A stored course is a relative object. If one or both of the points which define a stored course are redefined (coordinates changed), the course is moved (position, length, and direction are changed).

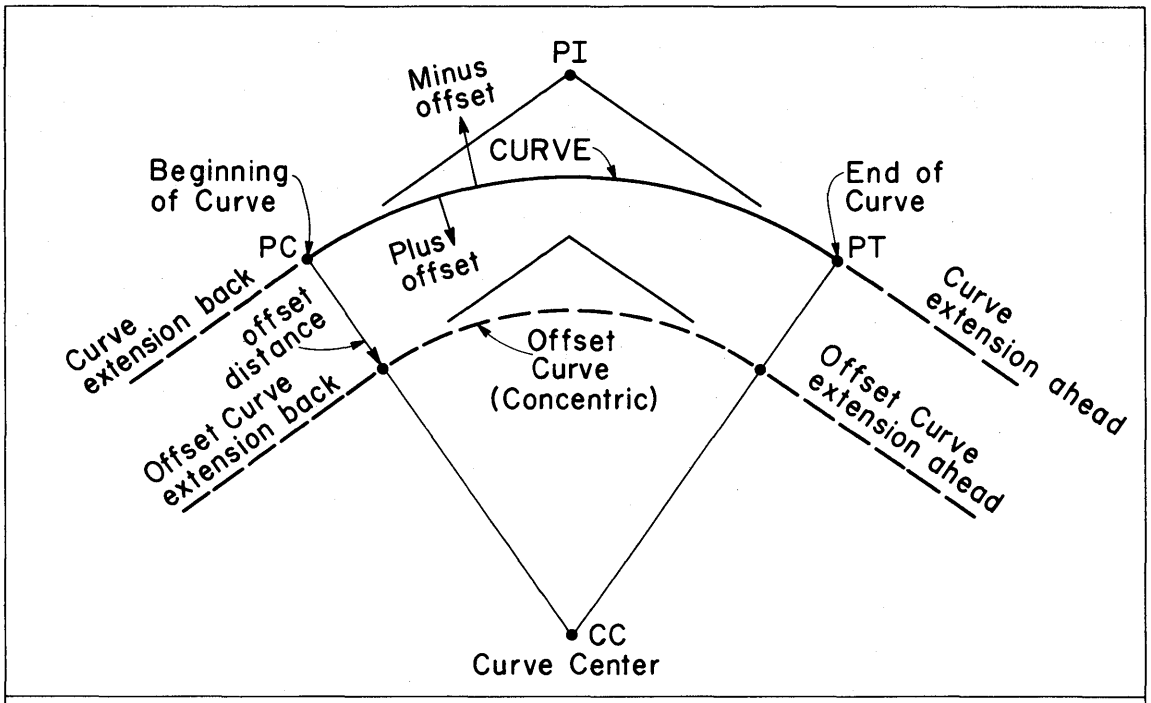
In operations such as intersecting two objects, one or both of which is a course, if the intersection does not occur on the course, the course is extended and treated like a line, and the user is so notified by a message.

Curve

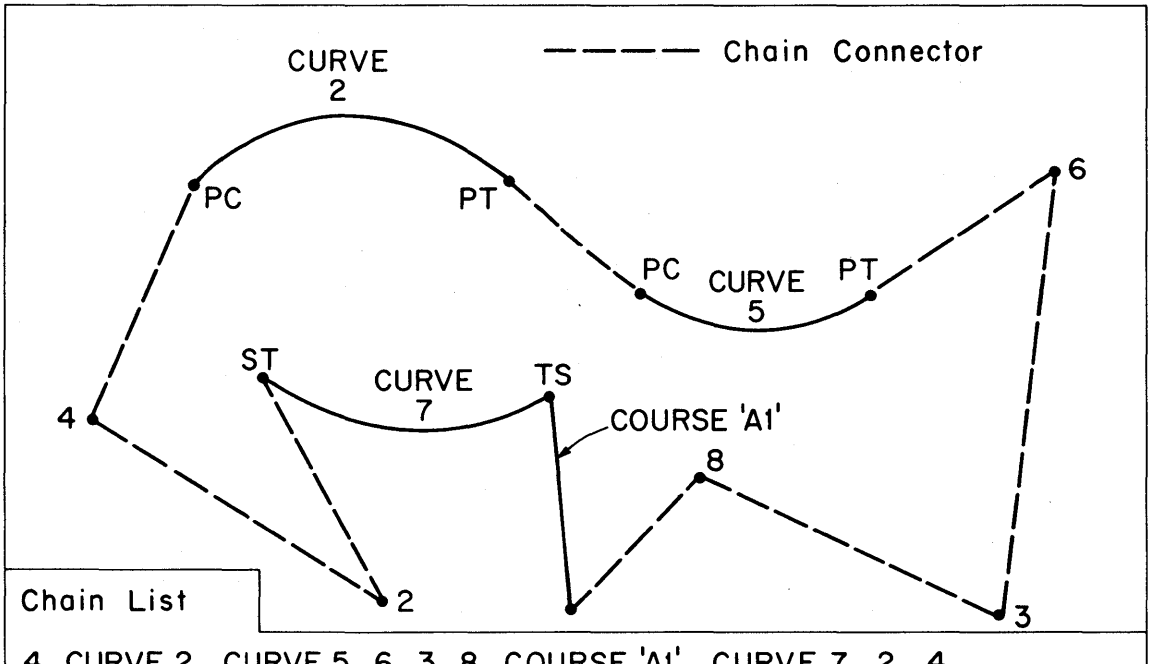
A curve is defined geometrically as a circular curve segment in the horizontal plane, with or without spiral transitions of equal or unequal length. A curve has specific end points (PC or TS and PT or ST) and absolute position in space. If a curve has transition spirals, the entire curve (circular curve segment plus spirals) is stored and retrieved as a single curve with a single identification number.

A curve is stored internally with numerical values which completely define its geometry and position in space. The station and horizontal coordinates of each curve point are stored in the Curve Table, as are the basic curve elements (radius, length, spiral lengths, and so forth). A curve is stored as an absolute object. Once stored, it is independent of the points and values initially used to define it. If any single curve point or curve element is to be redefined, the entire curve is redefined.

In operations such as intersecting two objects, one or both of which is a curve, if the intersection does not occur on the curve, the curve may be extended. Internally, curves are extended by extending the back and ahead tangents to the curve. Hence curve extensions are lines extending back from the beginning of the curve (PC or TS) and ahead from the end of the curve (PT or ST). Projections onto curve extensions are always to the nearest tangent.



CURVE OBJECT NOMENCLATURE



EXAMPLE OF CHAIN OBJECT

Chain

A chain is defined geometrically as a continuous linear or curvilinear object in the horizontal plane which can be defined in terms of stored points, courses, curves, and other chains. Objects such as alignments, traverses, parcels, baselines, and so forth, are classified as a single general object called a chain and are stored in the Chain Table. The term chain is derived from the concept of the chain object as a series of connected links, each link being a stored object which is to be connected with the previous object. Chains are defined and stored in the Chain Table as an ordered list of the links, the chain list.

A chain is a relative object, the chain list defining the topology of the chain. When a chain is used for purposes of computation or plotting, the geometry of chain is constructed from the current numerical values in the Point and Curve Tables for the objects in the chain list. If an object in the list is redefined (moved), the geometry of the chain will be different in subsequent references to it. Points in the list may be thought of as thumbtacks, and curves and courses may be thought of as templates with a thumbtack at each end. If one or more of the tacks are moved (coordinates changed), the chain takes on a new shape and position for that portion affected.

In constructing the geometry of a chain from the absolute values associated with the objects in the chain list, the internal processing routines generate a connector for each object in the list not preceded by the word GAP. The connector takes the form of a straight line segment (a course) which connects the object with the previous object in the list. The beginning point of the connector course is the end point of the previous object, and the end point of the connector course is the beginning point of the object. If, for example, two curves are being connected, the connector would be the course from the PT (or ST) of the previous curve to the PC (or TS) of the current curve. In chain computations, descriptions, and plots, the connectors are treated as courses which make up the continuous chain. A gap may be included in the chain by using the work GAP in the list between the two objects which are not to be connected. GAP means no connector course is to be generated.

No connector course is generated between the last object in the list and the first. Hence a chain is assumed to be an open chain, a gap occuring between the last and the first object. A closed chain can be obtained by making the end point of the last object coincide with the beginning point of the first object. For example, if the last object in the list is a stored point and it is the same point as the first object in the list, the chain will be closed on itself.

INPUT/OUTPUT CONTROL

System Settings

Facility is provided to modify the system for local surveying and geometric conventions. The Set command is used to set switches which control input/output conventions such as

a) N, E or X, Y coordinate system
b) Azimuths or bearings as standard direction
c) Azimuths measured from North or from South
d) Two or three decimal places on distance and coordinate output
e) Even seconds of arc or decimal seconds on output

The Set command is also used to control the mode of operation and to store constants.

There are standard settings which are fixed for a given installation. The system is automatically initialized with the standard settings each time the command COGO is executed. The Engineer can change the standard settings for his individual run by use of the Set command. Once a setting is made, it remains unchanged until another Set command is given which changes the setting. The Set command can be given at any point in a COGO problem.

The following are examples of the Set command:

```
SET SYSTEM  NE, BEARINGS, NAZIMUTH, DEC3, ASEC
SET SYSTEM  XY, AZIMUTHS, SAZIMUTH, DEC2, ADEC
```

Check Mode

The system operates in two modes: the Compute Mode, which is the normal mode of operation, and the Check Mode, which is a special mode of operation. When in the Compute Mode, entries are made in the data tables when commands are executed. When in the Check Mode, when commands are executed, the computed values for "redefined" points, scalars, and curves are not stored but are compared with the values already stored. If they do not compare within a specified tolerance, a NO CHECK message is printed along with the computed and the stored values. The stored values are left unchanged, and the system goes on to the next command.

The system is put into the Check Mode and taken out of the Check Mode (put into the Compute Mode) by the Engineer by use of the Set Command. The tolerance value is also specified by the Set command. The Check Mode provides the Engineer with a capability for checking a COGO solution via COGO commands. The Engineer can check his own solution or another Engineer's solution. Since each COGO problem can usually be solved in a variety of different ways, the Check Mode capability provides a very flexible and practical method of independent checking and verification.

Data Forms

Geometric variables such as distances, directions, and angles which are used as input data items in many commands can be expressed in the following basic forms:

a) Numerical Data Value

Examples:
```
DISTANCE 125.16
BEARING N 27 15 24.39 W
ANGLE 83 45
```

b) Stored Data Value

Examples:
```
DISTANCE 'X4'
AZIMUTH 'A'
ANGLE 'B3'
```

c) Computed Data Value

Examples:
```
DISTANCE POINT 4 TO POINT 8
BEARING PI 3 TO PC 5
ANGLE AT 4 FROM 6 TO 8
```

The computed form provides for rather extensive computations to be carried out within commands, making rather simple commands quite powerful in scope of application. Computed forms are also provided for such standard data items as station, offset, line, and course.

Output Control

Printed output is under command control; that is, the Engineer specifies the output he wants via commands. The Print command and the Describe command are used to print the numerical values associated with or computable from stored objects, such as the following examples:

```
PRINT POINTS 4, 8, 20 TO 60, 15, 400 TO 600
PRINT DISTANCES 4 TO 8, 5 TO 10 TO 8
PRINT ANGLE AT POINT 4 FROM 12 TO 16
DESCRIBE ALIGNMENT 'B3', PARCEL 'LOT/4', TRAVERSE 'T48'
PRINT ALL DISTANCES
PRINT ALL TABLES
```

Undefined objects in the print list are ignored when the command is executed. All output is under format control and is fully labeled. Each command is reproduced in the printed output; that is, the input is reproduced in the output. Intermediate output (output following the command execution) is provided for in many commands. Intermediate output may be generated automatically through the Set command or can be requested selectively as part of the command.

```
$
$ ..........................................................................
$    ....................  EXAMPLES OF STANDARD OUTPUT FORMATS  ..............
$ ..........................................................................
SET SYSTEM NE, BEA, NAZ, DEC3, ADEC

PRINT POINT 4
        POINT       4        N    540125.328     E    101050.875     S      5+ 0.0        Z     3259.640

PRINT DISTANCE 8 TO PC 24
    DISTANCE FROM       8 TO PC  24    5737.154

PRINT BEARING 7 TO 2
    BEARING FROM       7 TO       2  S 59  5 49.99 W

PRINT ANGLE AT 8 FROM 2 TO 4
    ANGLE AT       8 FROM       2 TO       4    275 37 25.37

PRINT LINE 35
    LINE  35 THRU       4  AT BEARING  N 45 39 44.88 E

PRINT COURSE 'SIDE'
    COURSE SIDE FROM       2 TO       7          LENGTH 32094.123          BEARING  N 59  5 49.99 E

PRINT CURVE 24

    CURVE  24    TYPE SCS     CURVE ELEMENTS

        TTB           2783.610  DEF.    109  4 30.47
        TTA           2832.812  BACK  N 49 57 40.49 W
        TOTAL LEN     3885.838  AHEAD N 59  6 49.99 E

      SPIRAL BACK

        LENGTH         200.000  ANGLE     3  0  0.00     XC     199.945
        LONG  TAN.     133.352  BACK  N 49 57 40.49 W   YC       3.490
        SHORT TAN.      66.684  AHEAD N 46 57 40.49 W   P        0.873
        LONG CHORD     199.976  PHI C     0 59 59.92     K       99.991

      CIRCULAR SECTION

        RADIUS        1909.859  DEGREE    3  0  0.00
        LENGTH        3385.838  DELTA   101 34 30.47
        TANGENT       2340.682  BACK  N 46 57 40.49 W
        EXTERNAL      1111.126  AHEAD N 54 36 49.99 E
        LONG CHORD    2959.546
        MID. ORD.      702.451
```

SPIRAL AHEAD

LENGTH	300.000	ANGLE	4	30	0.00	XC	299.815	
LONG TAN.	200.065	BACK	N 54	36	49.99 E	YC	7.851	
SHORT TAN.	100.059	AHEAD	N 59	6	49.99 E	P	1.963	
LONG CHORD	299.918	PHI C	1	29	59.72	K	-149.969	

CURVE 24 TYPE SCS CURVE POINTS

POINT CC	24	N	556140.124	E	108860.528	S	**********	Z	**********
POINT TS	24	N	554612.925	E	107707.898	S	216+85.862	Z	**********
POINT ST	24	N	557857.814	E	108007.828	S	255+71.699	Z	**********
POINT PI	24	N	556403.637	E	105576.739	S	244+69.472	Z	**********
POINT SR	24	N	554698.711	E	107605.802	S	218+19.214	Z	**********
POINT SC	24	N	554744.223	E	107557.063	S	218+85.862	Z	**********
POINT CI	24	N	556341.721	E	105846.277	S	242+26.543	Z	**********
POINT CS	24	N	557697.171	E	107754.560	S	252+71.699	Z	**********
POINT SA	24	N	557755.114	E	107836.135	S	253+71.758	Z	**********

DESCRIBE ALIGNMENT 'ROAD/3'

DESCRIPTION OF CHAIN ROAD/3

CHAIN ELEMENTS

COURSE	FROM 4	TO 8	LENGTH 15448.697	BEARING	N 45 39 44.88 E
COURSE	FROM 8	TO TS 24	LENGTH 5737.164	BEARING	N 49 57 40.49 W

CURVE 24 TYPE SCS CURVE ELEMENTS

TTB	2783.610	DEF.	109	4	30.47	
TTA	2832.812	BACK	N 49	57	40.49	W
TOTAL LEN	3885.838	AHEAD	N 59	6	49.99	E

SPIRAL BACK

LENGTH	200.000	ANGLE	3	0	0.00	XC	199.945	
LONG TAN.	133.352	BACK	N 49	57	40.49	W	YC	3.490
SHORT TAN.	66.684	AHEAD	N 46	57	40.49	W	P	0.873
LONG CHORD	199.976	PHI C	0	59	59.92		K	99.991

CIRCULAR SECTION

RADIUS	1909.859	DEGREE	3	0	0.00			
LENGTH	3385.838	DELTA	101	34	30.47			
TANGENT	2340.682	BACK	N 46	57	40.49	W		
EXTERNAL	1111.126	AHEAD	N 54	36	49.99	E		
LONG CHORD	2959.546							
MID. ORD.	702.451							

SPIRAL AHEAD

LENGTH	300.000	ANGLE	4	30	0.00	XC	299.815	
LONG TAN.	200.065	BACK	N 54	36	49.99	E	YC	7.851
SHORT TAN.	100.059	AHEAD	N 59	6	49.99	E	P	1.963
LONG CHORD	299.918	PHI C	1	29	59.72		K	-149.969

COURSE FROM ST 24 TO 7 LENGTH 29261.311 BEARING N 59 6 49.99 E

CHAIN POINTS

POINT	4	N	540125.328	E	101050.875	S	5+ 0.0	Z	3259.640
POINT	8	N	550922.175	E	112100.325	S	159+48.697	Z	**********

CURVE 24 TYPE SCS CURVE POINTS

POINT CC	24	N	556140.124	E	108860.528	S	**********	Z	**********
POINT TS	24	N	554612.925	E	107707.898	S	216+85.862	Z	**********
POINT ST	24	N	557857.814	E	108007.828	S	255+71.699	Z	**********
POINT PI	24	N	556403.637	E	105576.739	S	244+69.472	Z	**********
POINT SR	24	N	554698.711	E	107605.802	S	218+19.214	Z	**********
POINT SC	24	N	554744.223	E	107557.063	S	218+85.862	Z	**********
POINT CI	24	N	556341.721	E	105846.277	S	242+26.543	Z	**********
POINT CS	24	N	557697.171	E	107754.560	S	252+71.699	Z	**********
POINT SA	24	N	557755.114	E	107836.135	S	253+71.758	Z	**********

POINT	7	N	572878.619	E	133119.573	S	548+33.010	Z	**********

ERROR CHECKING

Each command is subjected to extensive checking by the processor. Errors which are detected are reported by messages which appear in the output. The following general cases may be identified:

Command Errors

Commands which are illegal, improperly constructed, incomplete, or in any way fail to conform with the command specifications are detected and reported. Such commands are not fully processed and are not executed. Each command is scanned and processed from left to right. Once an illegal condition is detected, processing ceases and the remaining (right hand) portion of the command is printed to assist in locating the illegal condition. The system then goes on to the next command. Hence only one (the first one detected) illegal condition is reported per command (there may be others in the unprocessed portion).

Undefined Objects

Commands which reference (attempt to retrieve and compute with) an undefined object (not stored in data table) are detected and reported. Such commands are not fully executed and may not be fully processed. The first undefined object in the command which the processor attempts to retrieve inhibits further processing and execution of the command. The undefined object is identified in the output. The undefined object reference counter is updated by one, and the system goes on to the next command. Hence only one (the first one detected) undefined object is reported per command (there may be others in the unprocessed and unexecuted portions).

Geometric Errors

Commands which call for computations which are not geometrically possible are detected and reported. An example would be an attempt to intersect two objects which do not intersect, even if extended, such as two concentric curves. The error message which is printed identifies the invalid condition. Further processing and execution of the command is inhibited, and the system goes on to the next command.

Geometric Warnings

Commands which call for computations which are possible but which require assumptions for execution are detected and reported. An example would be an attempt to intersect two objects which do not intersect themselves but which do intersect if one or both is extended. Whenever a reasonable assumption can be made which will result in a solution which may be acceptable to the Engineer, such action is taken. A warning message is printed which identifies the action taken. Such commands are processed and executed as valid commands. It is the Engineer's responsibility to review the output and satisfy himself that the action taken was valid. If not, he should resubmit the problem with corrected commands.

A careful study of the output from a COGO run should be standard practice. In addition to reviewing the error and warning messages, the Engineer should study the information included in the output to assist him in verifying whether or not his problem was correctly stated. Each problem should conclude with a group of commands under the Check Mode to help verify the solution. A random traverse through the problem is an excellent way to check the solution.

```
$ ..........................................................................

$    .....................      EXAMPLES OF TYPICAL MESSAGES      ..............

$ ..........................................................................

$ ..........................................................................

$    .....................    EXAMPLE OF COMMAND ERROR MESSAGE    ..............

$ ..........................................................................

LOCATE 50 FROM 4 OFFSET 50. ANGLE 20
**** INPUT ERROR 7.1 - REQUIRED DATA IS MISSING
**** INPUT WARNING 7.7 - AN ERROR WAS DETECTED WHILE PROCESSING THIS COMMAND.
**** INPUT WARNING 7.8 - COMMAND NOT COMPLETELY PROCESSED
**** SYMBOLS OF COMMAND INPUT NOT YET PROCESSED FOLLOW -
 ****ANGLE 20                                                          ****

$ ..........................................................................

$    .....................  EXAMPLE OF REDEFINED OBJECT MESSAGE  ..............

$ ..........................................................................

LOCATE 7 FROM 4 DIST 200. AZ 45
 POINT       7 BEING REDEFINED
```

```
$ ..........................................................................
$    .....................   EXAMPLE OF UNDEFINED OBJECT MESSAGE  ..............
$ ..........................................................................
LOCATE 60 FROM 4 DIST 4 TO 15, AZ 8 TO 20

*****************************************************************************

 ERROR. -   POINT'       15 HAS NOT BEEN STORED.

*****************************************************************************

 ERROR EXIT FROM THIS COMMAND

*****************************************************************************

**** INPUT WARNING 7.8 - COMMAND NOT COMPLETELY PROCESSED
**** SYMBOLS OF COMMAND INPUT NOT YET PROCESSED FOLLOW -
 ****AZ 8 TO 20                                                          ****
$ ..........................................................................
$    .....................   EXAMPLE OF GEOMETRIC ERROR MESSAGE   ..............
$ ..........................................................................
LOCATE 70, INTERSECT LINE THRU 4 TO 8 WITH LINE THRU 4 AT 45 15 20

*****************************************************************************


        ERROR - NO INTERSECTION.
 ERROR EXIT FROM THIS COMMAND

*****************************************************************************


$ ..........................................................................
$    .....................  EXAMPLE OF GEOMETRIC WARNING MESSAGE  ..............
$ ..........................................................................
LOCATE 80, PROJECT 7 ON CURVE 3

*****************************************************************************

 WARNING.  POINT     30 HAS BEEN LOCATED ON EXTENDED TANGENT TO CURVE.
           TANGENT HAS BEEN EXTENDED      413.535 FROM THE PT.

*****************************************************************************


$ ..........................................................................
$    .....................         EXAMPLE OF SET CHECK MODE        ..............
$ ..........................................................................
SET SYSTEM CHECK

SET CONSTANT DTOL 0.01

LOCATE 8 FROM 4, DIST 561.74, AZ 45 15 20
 **    CHECK**

LOCATE 8 FROM 4, DIST 561.0, AZ 45
 **NO CHECK**
    CALCULATED VALUES
         POINT       8        N   539680.862     E   172755.930     S ************     Z ************
    STORED VALUES
         POINT       8        N   539679.610     E   172758.221     S ************     Z ************
```

PROBLEMS, RUNS, and FILES

Problem

A COGO problem is defined as the ICES System command COGO followed by one or more COGO commands. A problem may have any number of COGO commands. A problem is terminated by another command COGO (initiating the next COGO problem) or by some other ICES System command such as another subsystem name or the command FINISH.

Problem Data Tables

A new set of COGO data tables is initialized each time the command COGO is given. Accordingly, each problem has its own set of data tables which is built up by the COGO commands for the problem. Provision is provided to save the problem data tables. If they are not saved, the problem data tables are automatically destroyed at the end of the problem.

Run

A COGO run is defined as one or more COGO problems submitted at the same time as a batch for processing on the computer. A run can have any number of COGO problems, each starting with the command COGO. A run is terminated by the ICES System command FINISH.

Files

Problem data tables can be saved, restored, deleted, listed, and printed with the FILE command. Each set of problem data tables to be saved and restored is given a unique file name. The problem data tables are stored as a permanent file on a secondary storage device under the filename. When the FILE SAVE command is given, the current contents of the problem data tables are filed. When the FILE RESTORE command is given, the problem data tables which were saved are restored and become part of the problem data tables for the current problem. All defined objects in the restored data tables are then available for use in the COGO commands which follow. The restored data tables are merged with the current data tables, the restored objects taking precedence.

Usually, the FILE SAVE command will be the last COGO command in a problem if the complete set of data tables is to be saved. However, it can be given at any point if only the contents of the data tables up to that point are to be saved. Similarly, the FILE RESTORE command will usually be the first COGO command (following the command COGO) if the problem is to start out with the previously saved tables. However, it can be given at any point if the saved data tables are not needed until that point.

Through the use of the FILE command and the save and restore capability, the same Engineer can work on the same problem over a period of time, continuing work on the problem in different runs. Or different Engineers can work on the same problem, each having access to the same problem data tables through use of a common filename.

```
$  $  $  $  $  $  $  $  $  $  $  $  $  $  $  $  $

$  $  $  $  $  $  $  $        RUN 1        $  $  $  $  $  $

$ ..........................................................................

$ ..............................     PROBLEM 1     ..............................

$ ..........................................................................

COGO                                    $    DATA TABLES INITIALIZED FOR PROBLEM 1
WELCOME TO COGO,YOUR FRIENDLY COORDINATE GEOMETRY SYSTEM

SET SYSTEM NE BEA

STORE 25 N 3000 E 2000 S 5+00

LOCATE 1 FROM 25 175.16 N 45 15 E OFFSET PLUS 50

STORE CURVE 8 PB 1 DB N 45 15 E TL 123.28 DEG 2 30 P LENGTH 3000
     STA OF PB STA OF 25 PLUS 175.16

STORE BEARING 'A' PI 3 TO PT 8 PRINT
    BEARING   A      S 59 45  0.00 E

LOCATE 12 FROM PT 8 256.38 'A'

STORE LINE 4 THRU 25 AT N 45 15 E

LOCATE 2 INTERSECT CURVE 8 WITH LINE 4 OFFSET PLUS 74

FILE SAVE 'PASSWORD'  'PROBLEM1'
 ****************************************************
 PROBLEM DATA TABLES SAVED IN FILE PROBLEM1
 ****************************************************

$ ..................    DATA TABLES FROM PROBLEM 1 SAVED    ...................

$ ..........................................................................

$ ..............................     PROBLEM 2     ..............................

$ ..........................................................................

COGO                                    $    DATA TABLES INITIALIZED FOR PROBLEM 2
WELCOME TO COGO,YOUR FRIENDLY COORDINATE GEOMETRY SYSTEM

SET SYSTEM NE BEA

STORE 46 N 2500 E 1000

D     18 N 1000 E 3500

TRAVERSE 'T5'

BACK 46 TO 18

COURSE 46 TO 32 DIST UNKNOWN  ANGLE MINUS 77 0

D            25      2500           M      98
```

```
L            78       3150      ANGLE UNKNOWN

D            52       3620            M      56

D            18       UNKNOWN         M      97

AHEAD 18 TO 46 CANGLE MINUS 157 0 0

END OF TRAVERSE
*****************************************************************************
 CLOSED TRAVERSE
*****************************************************************************
    UNKNOWN ANGLE AT POINT    25  CLOCKWISE  125  0  0.00
    UNKNOWN NOT COUNTED AS A MISSING PART
 TWO MISSING DISTANCES
    DISTANCE FROM POINT    46    TO POINT    32  2716.208
    DISTANCE FROM POINT    52    TO POINT    18  3485.710
*****************************************************************************

$ ..........................................................................

$ ..........................      PROBLEM 3       ..........................

$ ..........................................................................

COGO
WELCOME TO COGO,YOUR FRIENDLY COORDINATE GEOMETRY SYSTEM

SET SYSTEM NE BEA

FILE RESTORE 'PROBLEM1'
 ***************************************************

 FILE PROBLEM1 STORED IN DATA TABLES

 ***************************************************

$ ...................  DATA TABLES FROM PROBLEM 1 RESTORED   ...................

LOCATE 5 PROJECT 2 ON LINE 4 OFFSET P 50

D       3 INTERSECT LINE 4 OFFSET P 74 WITH LINE THRU PI 8 AT 'A'

D       4 PROJECT 3 ON CURVE 8

D       8 INTERSECT CURVE 8 OFF P 24 WITH COURSE PC 8 TO 12

PRINT POINTS 1 TO 30
          POINT       1        N      3087.806     E      2159.597     S ************     Z ************
          POINT       2        N      3390.444     E      2498.978     S     11+30.405      Z ************
          POINT       3        N      4400.149     E      3517.532     S ************     Z ************
          POINT       4        N      3822.410     E      3587.006     S     23+14.161      Z ************
          POINT       5        N      3407.488     E      2482.081     S ************     Z ************
          POINT       8        N      3177.367     E      2284.284     S ************     Z ************
          POINT      12        N      3397.580     E      5236.666     S ************     Z ************
          POINT      25        N      3000.000     E      2000.000     S      5+ 0.0        Z ************

PRINT CURVE 8

    CURVE    8     TYPE C        CURVE ELEMENTS

          RADIUS          2291.831  DEGREE    2 30  0.00
          LENGTH          3000.000  DELTA    75  0  0.00
          TANGENT         1758.584  BACK   N 45 15  0.00 E
          EXTERNAL         596.959  AHEAD  S 59 45  0.00 E
          LONG CHORD      2790.357
          MID. ORD.        473.599


    CURVE    8     TYPE C        CURVE POINTS

          POINT CC    8        N      1546.972     E      3860.631     S ************     Z ************
          POINT PC    8        N      3174.597     E      2247.148     S      7+98.440      Z ************
          POINT PI    8        N      3526.737     E      5015.196     S     37+98.440      Z ************
          POINT PI    8        N      4412.666     E      3496.069     S     25+57.024      Z ************


FILE SAVE 'PASSWORD'  'PROBLEM3'
 ***************************************************
 PROBLEM DATA TABLES SAVED IN FILE PROBLEM3
 ***************************************************

$ ...................    DATA TABLES FROM PROBLEM 3 SAVED    ...................

$ ..........................................................................

$ ..............    THE NEXT COMMAND SPECIFIES THE END OF RUN 1   ..............

$ ..........................................................................

$    $    $    $    $    $    $    $    $    $    $    $    $    $    $    $

FINISH
```

ENGINEERS' GUIDE

to

ICES COGO I

* * * * * * * * * * * * *

PART 2

COMMAND SPECIFICATIONS

Civil Engineering Systems Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

SECTION DIRECTORY

COMMAND DIRECTORY

SUMMARY OF DATA TABLES

Object Label	Id. Number or Name**	Table Capacity	Table Entries	Absolute or Relative
Point POINT	Numeric 1 to 4 digits 0 to 9999	10,000 Points	Horizontal Coord. Station Elevation	Absolute
Distance DISTANCE	Alphanumeric 1 to 4 characters	Open*	magnitude (Absolute value)	Absolute
Angle ANGLE	Alphanumeric 1 to 4 characters	Open*	magnitude (Absolute value)	Absolute
Direction AZIMUTH BEARING	Alphanumeric 1 to 4 characters	Open*	magnitude (Absolute value)	Absolute
Line LINE	Numeric 1 to 3 digits 0 to 999	1,000 Lines	Point Number Direction	Relative Absolute
Course COURSE	Alphanumeric 1 to 4 characters	Open*	Beginning and End Point Numbers	Relative
Curve CURVE	Numeric 1 to 3 digits 0 to 999	1,000 Curves	Coordinates and Station of all Curve Points. Curve Elements	Absolute
Chain CHAIN TRAVERSE ALIGNMENT PARCEL	Alphanumeric 1 to 8 characters	Open*	Chain List	Relative
Profile PROFILE	Alphanumeric 1 to 8 characters	Open*	VPI Points VC lengths	Relative Absolute

* Capacity a function of hardware configuration but not a practical constraint.

** Alphanumeric names must be enclosed in single quotes.

COMMAND CONVENTIONS and STANDARD DATA ITEMS

Command Description Conventions

In order to describe the specifications for the commands in concise form, a set of standard conventions, rules, symbols, and data items has been adopted and is described herein. A typical command might be specified in the following sample form:

LOCATE n INTERSECT object with object (NEAR pa) print

In this sample command the conventions provide the following interpretations:

1) The word LOCATE, the first word, is the command name. The minimum abbreviation is LOC, the underscored portion.

2) The symbol n stands for the identification number of an integer point to be stored. Since it is underscored, it is required.

3) The word INTERSECT is a command modifier, i.e., it denotes which method for point location is to be performed. It is required and the minimal abbreviation is INT.

4) The lower case words "object" are data items which can take any of the standard allowable forms. Since they are underscored, they are required. If the command were such that either or both were not required, they would not be underscored.

5) The word "with" may be included to make the command more readable. It is one of a number of words which, if they appear in a command specification in lower case, are ignored in command processing. Such "ignorable words" must be fully written, i.e., they cannot be abbreviated.

6) The data item (NEAR pa) is optional because it is enclosed in parentheses. If it is included, the word NEAR and the symbol pa must be given. The symbol pa stands for the identification number of any stored point (or curve point of any stored curve).

7) The word "print" stands for an optional (because it is not underscored) command modifier.

The following are actual commands which could follow from the above general form:

```
LOCATE 4 INTERSECT LINE 1 WITH CURVE 5 NEAR PC 5 PRINT
LOC     5 INT      LINE THRU 4 AT AZI 45 LINE THRU 2 TO 7
LOC    10 INT      CUR 7 CUR 3 N PI 3 PRI
```

Even though upper case and lower case are used throughout the command descriptions, they are so used only to differentiate between standard data items, objects, command modifiers, and command names. An actual command, when written and keypunched, must have all characters in upper case and must have one or more blanks (or a comma) separating all words, items, and values within the actual command.

Command Ditto

If the previous command is to be repeated, the command word may be replaced by a D (for ditto) followed by one or more blanks. The ditto feature is used for STORE in the following example commands:

```
STORE POINT 4  100  200
D     DISTANCE 'X' 525.16
D     ANGLE 'A1' AT 4 FROM 6 TO 8
```

Command Continuation

If a command requires more than one line (card), it may be continued on the next line (card) by using a freestanding minus sign (one preceded and followed by one or more blanks). There may be up to 5 continuation lines (cards), with the maximum length for a command being 400 characters including blanks and comments.

Comments

Comments may be inserted by using the $ character (followed by one or more blanks) as the first character on the line (card). Comments may not be continued. If the comment requires more than one line (card), each comment line (card) should begin with the $ character.

If a command or command continuation requires less than a full line (card), the remaining unused portion may be used for comment information, the comment being separated from the command by a freestanding $ character.

Comments are not processed, but they are duplicated in the output as they are read. Liberal use of comments is encouraged.

Ignorable Words

In many instances in commands, simple words are allowed to make the command more readable but are optional and are ignored by the processor. The following is a list of such words:

on	from	of	with	rule
to	at	thru	by	

When they are shown in a command specification in small letters, they are optional and are ignored. If the Engineer includes them in a command, the full word must be given, not an abbreviation. In some cases such words are required words, modifiers, or data items label. In such cases the word is given in caps and is underscored.

Numerical Data Values (v, va, vb . . .)

Where a numerical data value is called for in a command or data item, the symbols v, va, vb . . . are used. Decimal points may be omitted for whole number data values, the following being equally acceptable:

200.0 200 200.

Decimal points should not be used for integer values such as point numbers.

In data items which include or are preceded by the PLUS/MINUS operator, a negative numerical value must be indicated by the MINUS modifier, not by a negative sign. With the exception of numerical coordinate and elevation values and other indicated exceptions, the system is based on numerical data values always being positive.

When a numerical data value for a bearing is used, the system expects natural quadrant labels such as in the following examples:

N 35 25 15 W S 45 E

When a numerical data value for a station is used, a plus sign may be included, but it must be embedded in the numerical value; that is, not preceded or followed by blanks. The following examples are acceptable:

5+00. 500 25+35.16 2535.16

A numerical angle or direction value can be given in any one of the following forms:

degrees, minutes, seconds	examples: 35 15 25.16 328 10 20
degrees, minutes	examples: 86 15.4 18 30
degrees	examples: 45.5 62

Algebraic Operations

Limited algebraic operations are permitted within some commands and data items. They are indicated by the appearance of the following:

p/m	The word PLUS or the word MINUS is to be given. If p/m is optional and is not given, PLUS is assumed.
operator	One of the following words is to be given to denote an algebraic operation: PLUS, MINUS, MULTIPLY (BY), or DIVIDED (BY)

Standard Symbols for Object Identifiers

The standard symbols for object identifiers used in command descriptions and their meaning are as follows:

Symbols	Meaning
n, na, nb . . .	Identification number of an integer point to be stored in the Point Table. The optional label (POINT) may precede the point number.
pi, pj, pk . . .	Identification number of an integer point already stored in the Point Table. The optional label (POINT) may precede the point number.
pa, pb, pc . . .	Identification number of a stored integer point or a stored curve point. The optional label (POINT) may precede an integer point number.
i, j, k . . .	Identification number of a line or curve, as determined by the preceding LINE or CURVE label.
a, b, c . . .	Name of a distance, angle, direction, course, chain, profile, or text, preceded by a required or optional label. The name must always be enclosed in single quotes.

Standard Allowable Forms for Scalar Data Items

Standard allowable forms for scalars (single valued quantities) appearing as data items in commands are as follows:

Scalar Data Item	Allowable Forms
distance	numerical value: (DISTANCE) v stored value : (DISTANCE) a computed value : (DISTANCE) from pa TO pb
angle	numerical value: (ANGLE) v stored value : (ANGLE) a computed value : (ANGLE) AT pa from pb to pc (Clockwise angle from pb to pc is computed)
c/angle	A positive clockwise angle is to be given, directly or indirectly. It may be given directly in the form PLUS angle or indirectly in one of the following forms MINUS angle The value of angle is subtracted from 360° to obtain the clockwise angle. P/M DEFLECTION angle If PLUS, the value of angle is added to 180° to obtain the clockwise angle. If MINUS, the value of angle is subtracted from 180° to obtain the clockwise angle.
direction	numerical value: (AZIMUTH) v, (P/M angle) stored value : (AZIMUTH) a, (P/M angle) computed value : (AZIMUTH) pa TO pb, (P/M angle) line direction : (AZIMUTH) of LINE i, (P/M angle) The optional word (BEARING) may be used in place of the optional word (AZIMUTH). If the optional data item (P/M angle) is included, the given value is modified by the value of angle to obtain the direction to be used for the direction data item.
station	numerical value: (STATION) v (P/M distance) stored value : (STATION) OF pa (P/M distance) If the optional data item (P/M distance) is given, the given value is modified by the value of distance to obtain the station to be used for the station data item.
offset	standard form: OFFSET P/M distance PLUS means to the right in the forward sense implied by the command in the horizontal plane or up in the vertical plane. MINUS means to the left or down.

Standard Allowable Forms for Object Data Items

Standard allowable forms for the following types of objects appearing as data items in commands are as follows:

Object Data Item	Allowable Forms
line	LINE i, offset (i is a stored line) LINE THRU pa at direction, offset LINE THRU pa TOWARD pb, offset
course	COURSE a, offset (a is a stored course) COURSE from pa TO pb, offset
curve	CURVE i, offset (i if a stored curve)
chain	CHAIN a, offset (a is a stored chain) TRAVERSE, ALIGNMENT, BASELINE, or PARCEL may be used in place of the word CHAIN

In each case, if the optional data item offset is included, the object actually used in the computations will be one parallel to and/or concentric with the given object at the offset distance.

Examples of Allowable Forms for Data Items

Data Item	Example
distance	125.175 DISTANCE 1000 DIST 'A' 4 TO 8 DIST FROM POINT 12 TO PC 5
angle	90 42 15 55.93 ANGLE 'A25' 'A25' AT 4, 8, 3 ANGLE AT POINT 12 FROM 4 TO PC 5
direction	90 N 25 15 30.6 E AZIMUTH 'AZ4' PLUS ANGLE AT 5, 7, 2 14 TO 8 PLUS 90 AZ PC 5 TO PI 5 MINUS ANGLE 'A25' PC 5 TO PI 5 M 'A25' AZ OF LINE 28 M 90

Data Item	Example
course	COURSE 'M' COU 'C25', OFFSET PLUS 50 COURSE 'JACK', OFFSET MINUS DISTANCE FROM PC 3 TO PT 8 COURSE FROM POINT 4 TO POINT 7 COU 4 TO 7, OFF P 25
curve	CURVE 2 CUR 4, OFF PLUS DIST 6 to 3765
chain	CHAIN 'F' TRAVERSE 'J-W' ALIGN 'ROUTE-3', OFFSET MINUS DIST 20 TO 30 PARCEL 'JONES' BASELINE 'B', OFFSET PLUS 125.
station	2+00 STA 8+00. STA 125+00 PLUS DIST FROM 4 TO 7 STA OF PC 5 MINUS 10. STA OF POINT 4 PLUS PC 3 TO PI 3 STA OF 2
offset	OFFSET PLUS 125.12 OFF M 100 OFFSET PLUS DIST FROM POINT 15 TO POINT 4 OFF M 3 TO 8
line	LINE 4 LINE 4, OFFSET PLUS 100. LINE THRU 2 AT N 35 15 20 W, OFFSET MINUS PC 3 TO 18 LINE THRU 5 AT AZ 2 TO 7 PLUS ANGLE 'A4', OFF P DIST 'X' LINE THRU 2 AT 270 LINE THRU 8 TOWARD 20 LINE THRU 7 AT 3 TO 5 PLUS 90, OFFSET PLUS 100

Print Modifier

The word print as an optional data item at the end of a command takes the form PRINT. If the print modifier is given, intermediate output will be printed, usually the data values for the object being stored by the command. The Set command can be used to automatically insert a print modifier wherever it is allowed.

SET COMMANDS

Set System

SET (SYSTEM) specs

where specs is one or more of the following (for each pair, only one can be given):

NE or XY
NAZIMUTH or SAZIMUTHS
AZIMUTHS or BEARINGS
DEC3 or DEC2
ASEC or ADEC
COMPUTE MODE or CHECK MODE
PRINT MODE or NOPRINT MODE
REDEFINE MODE or NORDEFINE MODE

The meaning of the settings is as follows:

NE	- On I/O, horizontal coordinates, North followed by East
XY	- On I/O, horizontal coordinates, X followed by Y
NAZIMUTHS	- On I/O, all azimuth values measured from north
SAZIMUTHS	- On I/O, all azimuth values measured from south
AZIMUTHS	- On output, directions printed as azimuths
BEARINGS	- On output, directions printed as bearings
DEC3	- On output, distance and coordinate values with 3 decimals
DEC2	- On output, distance and coordinate values with 2 decimals
ASEC	- On output, angle and direction values with even seconds
ADEC	- On output, angle and direction values with 2 decimals
COMPUTE	- Entries made in data tables
CHECK	- Comparisons made with data table entries
PRINT	- Optional print modifier automatically inserted for intermediate output after each command
NOPRINT	- No intermediate output unless requested by giving optional print modifier in command
REDEFINE	- Redefinition of objects is permitted and is not to be considered an error
NORDEFINE	- Redefinition of objects is not permitted, and if attempted, the command is considered in error and is not executed.

Set Constants

SET (CONSTANT) values

where values is one or more of the following:

DTOLERANCE v
ATOLERANCE v
MAXIMUM (ERRORS) v

The meaning of the various values forms is described in the sections which follow.

DTOLERANCE

The constant is for giving the tolerance to be used when the system is operating in the check mode when checking stored values for distances and coordinates. The value for v is the tolerance. For example, the command

SET CONSTANT DTOL 0.01

would mean that distances, coordinates, stations, and so forth, which compare within ± 0.01 would be considered "CHECK."

ATOLERANCE

The constant is for giving the tolerance to be used when the system is operating in the check mode when checking stored values for angles and directions. The value for v is the tolerance in seconds. For example, the command

SET CONSTANT ATOL 0.5

would mean that angles and directions which compare within ± 0.5 seconds would be considered "CHECK."

MAXIMUM ERRORS

The constant is for stating the maximum number of commands (except Print commands) which reference one or more undefined objects allowed in a single run before execution of further commands is inhibited. For example, the command

SET CONSTANT MAXIMUM ERRORS 10

would mean that after 10 such commands have been detected in a run, the commands which follow will not be executed.

FILE COMMANDS

File Save

FILE SAVE password filename

where password is a 1- to 8-character password defined by the installation in the INSTALLATION SET command and filename is a 1- to 8-character name to be assigned to the file which is to be created. Both password and filename must be enclosed in single quotes.

This command will cause the current contents of the COGO data tables (i.e., all presently defined objects) to be stored as a permanent file on disk, which file will have name filename.

The password prevents unauthorized users from creating files; i.e., if the given password is not one of those specified by the installation in the INSTALLATION SET command, then the file will not be created and a message will be printed.

If a file with filename already exists, a message will be printed, the old contents of the file destroyed, and the new contents stored in the file.

Example:

```
FILE SAVE 'USERA' 'PROBLEM1'
```

File Restore

FILE RESTORE filename

where filename is a 1- to 8-character name of the file whose contents are to be stored in the COGO data tables. This command has no effect on the file itself; i.e., the contents of the file will not be changed.

An automatic SET SYSTEM REDEFINE MODE command is executed internally for all redefinition of objects. If objects having the same identifiers appear both in the data tables and on the file, the value on the file will take precedence and will replace the value in the data table. The over-all effect of this command is to merge the file with the current contents of the data tables, the file taking precedence over the data tables.

If the system is in the CHECK mode when the FILE RESTORE command is given, the precedence rule will be reversed for the types of objects which are checked. The objects being read in from the file will be checked against the stored objects. If they do not check, the file values and the stored values are printed and the stored values are not changed.

Example:

```
FILE RESTORE 'PROBLEM1'
```

File Delete

FILE DELETE password list
FILE DELETE password ALL

where password is a 1- to 8-character password defined by the installation in the INSTALLATION SET command, list is a list of 1 or more file names which are 1 to 8 characters, and ALL specifies all files.

This command will cause the specified files to be deleted if the password is valid. If it is not valid, a message will be printed.

Password and the filenames in list must be enclosed in single quotes.

Examples:

```
FILE DELETE 'USERA' 'PROBLEM1' 'PROBLEM2'
FILE DELETE 'USERB' ALL
```

File List

FILE LIST

This command will list the names of all currently defined files in the COGO user data set.

File Print

FILE PRINT list
FILE PRINT ALL

where list and ALL are the same as in the file delete command. If the list form is used, each file name must be enclosed in single quotes.

This command will print the names and/or id's of all objects stored in the specified file name or in all files if the ALL form is used. No values will be printed, only the object identifiers.

Example:

```
FILE PRINT 'PROBLEM1' 'COGO37'
```

PRINT AND DELETE COMMANDS

The Print command provides for the printing of numerical data for stored objects and the Delete command the deletion of stored objects from the data tables. The general forms of the commands are

PRINT type list	DELETE type list
PRINT ALL types	DELETE ALL types
PRINT ALL TABLES	DELETE ALL TABLES

where type may be any one of the following words, and types may be more than one (TABLES is the equivalent of giving all types):

POINTS	AZIMUTHS	DISTANCES	PROFILES
LINES	BEARINGS	COURSES	TEXTS
CURVES	ANGLES	CHAINS	

and where list is one or more object names (a, b, c . . .) or identification numbers (i, j, k . . .). If an object in the list is not a stored object, a message will be printed to identify the undefined object, but it will not contribute to the error count. Further, the remaining objects in the list will be processed and printed.

When type is POINTS, LINES, or CURVES, the items in the print list can be of the form j TO k (meaning all stored objects in the identification number range j to k). When type is DISTANCES, AZIMUTHS, BEARINGS, or COURSES, the items in the list can take the form pa TO pb, and in the case of ANGLES, the form AT pa from pb to pc, the numerical output values being computed using the stored point values.

More detailed information on the print commands, including a description of the numerical values which are printed and special forms of the print command, is given under the command specifications for each type object.

The PRINT command word can also be abbreviated to the single letter P followed by one or more blanks.

STORE COMMANDS

The Store command provides for the defining of objects to be stored and the storage in the data tables of numerical values associated with such objects. The following is a summary list of such commands. They are defined in detail in the command specifications for each type object.

Point

STORE (POINT) n X v, Y v, STA v, Z v
STORE (POINT) n N v, E v, STA v, Z v

Distance

STORE DISTANCE a distance (operator, distance), print
STORE DISTANCE a RADIUS of CURVE i at station, print
STORE DISTANCE a STATION OF pa MINUS STATION OF pb, print

Angle

STORE ANGLE a angle (operator, modifier), print

Azimuth or Bearing

STORE AZIMUTH a direction, round, print
STORE AZIMUTH a TANGENT to CURVE i at station, print

Line

STORE LINE i thru pa at direction, print
STORE LINE i thru pa TOWARD pb, print
STORE LINE i thru n PARALLEL to LINE j, offset, print
STORE LINE i thru n TANGENT to CURVE j, at station, offset, print

Course

STORE COURSE a pa to pb, print
STORE COURSE a pa to n, distance, direction, print
STORE COURSE a na to nb, PARALLEL to COURSE b, offset, print

Curve

STORE CURVE i, reference, b/spiral, element, a/spiral, a/tangent, c/station
STORE CURVE i, CONCENTRIC with CURVE j, offset, c/station
STORE CURVE i, PARALLEL to CURVE j, offset, c/station

Chain

STORE CHAIN a, list

Text

STORE TEXT a n

The STORE command word can also be abbreviated to the single letter S followed by one or more blanks.

Store Point

STORE (POINT) n X v, Y v, STA v, Z v
STORE (POINT) n N v, E v, STA v, Z v

where n is the identification number of the point to be stored. If the numerical data values (v's) are in proper order, compatible with the system setting, the labels (X, Y, etc.) may be omitted. If one item has a label, all those which follow must have a label. If all items are labeled, they may be in any order. Assuming the system is set for X, Y, the following are examples of consistent commands.

```
STORE POINT 4  X 1000  Y 2000  STA 15+00  Z 300
STORE POINT 4    1000    2000      15+00    300
STORE POINT 4    1000    2000  Z 300
STORE POINT 4  N 2000  E 2000  Z 300  STA 15+00
STORE POINT 4  STA 15+00, Z 300
STORE POINT 4    1000    2000
STORE POINT 4  STA 15+00
```

Coordinates and elevation values may be entered with a negative sign.

Print Stored Points

PRINT POINTS pi, pj, pk . . . .
PRINT POINTS pi TO pj, (pk TO pl), ...
PRINT ALL POINTS

The first two forms may be mixed. The values printed are the table entries for the stored points. Undefined entries are printed as a row of asterisks.

Delete Stored Points

Same forms as for Print Stored points with use of word DELETE instead of PRINT.

Equate Point

EQUATE n to pa, print

where n is the identification number of a new point to be stored in the Point Table with the same data values as in the Point Table for stored point pa. The expected use of this command is for cases where a position in the horizontal plane has a number of different station and/or elevation values, depending on which alignment or profile it is associated with. Typically, this command would be followed by appropriate commands to store the correct station and elevation, preserving the horizontal coordinate values.

DISTANCE COMMANDS

Store Distance

STORE DISTANCE a distance, (operator, modifier), print

where a is the name of the distance to be stored. If the optional data items operator, modifier are included, the value of distance is modified by the value of modifier to compute the value to be stored for a. If operator is P/M, modifier takes any of the standard forms for distance; that is, facility is provided to add or subtract two distances. If operator is MULTIPLY or DIVIDE, modifier must be a numerical data value (v) which is treated as a scalar. For example,

STORE DISTANCE 'X' DIST 4 TO 8 PLUS 50.

The distance from point 4 to point 8 would be computed, 50. would be added to it, and the sum stored for 'X'.

Compute Radius and Store Distance

STORE DISTANCE a RADIUS of CURVE i at station, print

The radius to Curve i at the value of station is computed and stored. If station is omitted and i is a circular curve or a circular curve with transition spirals, the circular curve radius is stored. If station is omitted and i is a spiral, the command is considered to be an error and no value is stored. If station is specified but is not on the curve, the command is considered to be in error and no value is stored. In each such case, a message in printed.

Compute Station Difference and Store Distance

STORE DISTANCE a STATION OF pa MINUS STATION OF pb, print

The station of stored point pb is subtracted from the station of stored point pa and the difference (absolute value) is stored for a. A numerical value for station may be inserted in place of the OF pa and OF pb data items.

Print Stored Distances

PRINT DISTANCES a, b, c, ...
PRINT ALL DISTANCES

The value printed is the magnitude of the distance.

Compute and Print Distances

PRINT DISTANCES pa TO pb, (pc TO pd), ...
PRINT DISTANCES pa TO pb (TO pc TO pd . . . .

The two forms can be combined, such as

PRINT DISTANCES 2 TO 4, 6 TO 8 TO 7, 3 TO 5, PC 2 TO 9 TO PT 3

Delete Stored Distances

DELETE DISTANCES a, b, c . . . .
DELETE ALL DISTANCES

Example Distance Commands

```
STORE DISTANCE 'A' POINT 8 TO POINT 5
STORE DISTANCE 'B35' PC 3 TO CC 3 DIVIDE BY 2.0
STORE DISTANCE '99' DIST 'A' MINUS DIST 'B'
STORE DISTANCE 'JACK' 4 TO 6 PLUS 8 TO 10, PRINT

STORE DISTANCE 'M' RADIUS OF CURVE 4 AT STA 5+25.17
S DIS 'M' RAD CUR 4 5+25.17

STO DIST 'BEN' STA OF PC 18 MINUS STA OF PT 15

PRINT DISTANCES 'A', 'B35', '99', 'JACK', 'M', 'BEN'
PRI DIST 2 TO 4 TO 6 TO 8, 5 TO 7
```

ANGLE COMMANDS

Store Angle

STORE ANGLE a angle (operator, modifier), print

where a is the name of the angle to be stored. If the optional data items operator, modifier are included, the value of angle is modified by the value of modifier to compute the value to be stored for a. If operator is P/M, modifier takes any of the standard allowable forms for angle; that is, facility is provided to add or subtract two angles. If operator is MULTIPLY or DIVIDE, modifier must be a numerical data value (v) which is treated as a scalar. Examples:

```
STORE ANGLE 'A1' ANGLE AT 2 FROM 4 TO 6 PLUS ANGLE 'B'
STORE ANGLE 'C' ANGLE 'A1' DIVIDED BY 3.0
```

Printed Stored Angles

PRINT ANGLES a, b, c, ...
PRINT ALL ANGLES

The value printed is the magnitude of the angle in degrees, minutes, and seconds.

Compute and Print Angles

PRINT ANGLE AT pa from pb to pc, (AT pd from pc to pf), ...

Delete Stored Angles

DELETE ANGLES a, b, c
DELETE ALL ANGLES

DIRECTION COMMANDS

In the command specifications which follow, the word BEARING may be substituted wherever the word AZIMUTH appears.

Store Azimuth

STORE AZIMUTH a direction, round, print

where a is the name of the direction to be stored. If the optional data item round is included, it takes one of the following forms:

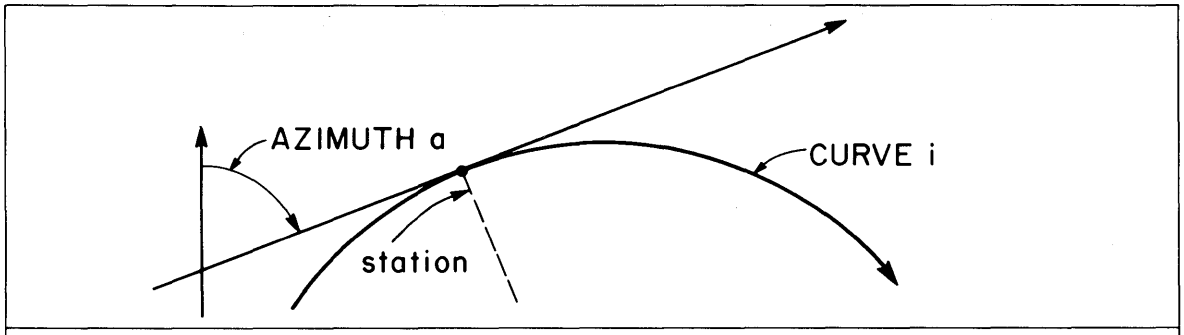
ROUND to v MINUTES
ROUND to v SECONDS

The value of direction is rounded to the nearest even v minutes or seconds before being stored.

Compute Tangent and Store Direction

STORE AZIMUTH a TANGENT to CURVE i at station, print

The direction of the tangent ahead at the value of station on curve i is computed and stored. If the station occurs before the beginning of the curve, the back tangent direction is stored; if after the end of the curve, the ahead tangent direction is stored. Messages are printed in these cases.



Print Stored Directions

PRINT AZIMUTHS a, b, c . . . .
PRINT ALL AZIMUTHS

The value printed is the magnitude of the direction in degrees, minutes, and seconds.

Compute and Print Directions

PRINT AZIMUTHS pa TO pb, round
PRINT AZIMUTHS pa TO pb, (pc TO pd), ...
PRINT AZIMUTHS pa TO pb (TO pc TO pd . . . .

The second and third forms may be mixed. In the first form the optional data item round is as described under Store Azimuth except that the rounding occurs before printing instead of before storage.

Delete Stored Directions

DELETE AZIMUTHS a, b, c . . . .
DELETE ALL AZIMUTHS

Print Direction of Stored Line

PRINT AZIMUTH of LINE i

Example Direction Commands

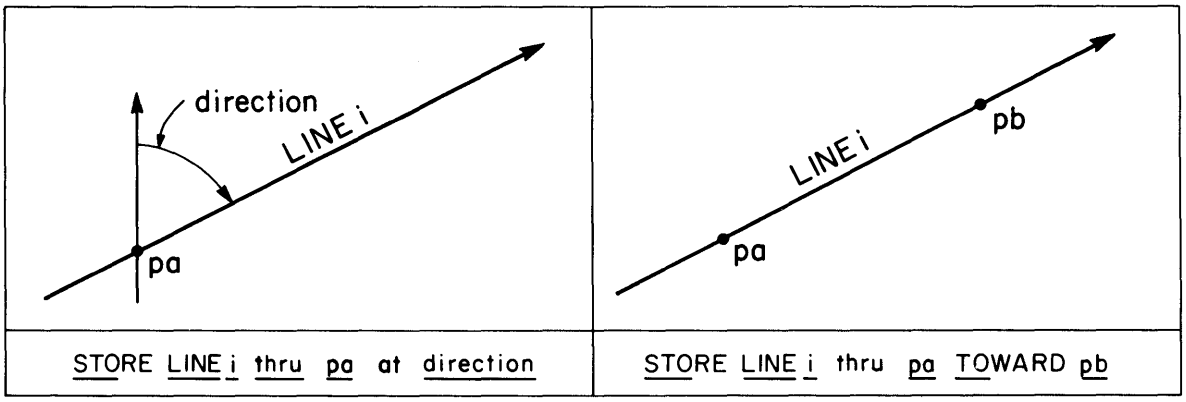
```
STORE AZIMUTH 'AZ1' 275 15 23.17
STORE AZIMUTH '4'    AZ 32 TO 56 PLUS 90
STORE AZIMUTH 'C3'   3 TO 7 PLUS ANGLE AT 15, 18, 20
STORE BEARING 'B'    5 TO 8 ROUND TO 30 SECONDS
STORE BEARING 'H'    TANGENT TO CURVE 4 AT STA 10+50
PRINT AZIMUTHS 2 TO 4, 6 TO 8, 3 TO 5 TO 7
```

LINE COMMANDS

Store Line

STORE LINE i thru pa at direction, print
STORE LINE i thru pa TOWARD pb, print

where i is the identification number of the line to be stored, pa is the stored point on the line, and direction is the direction of the line. In the second form, the direction of the line is computed as the direction from pa to stored point pb.

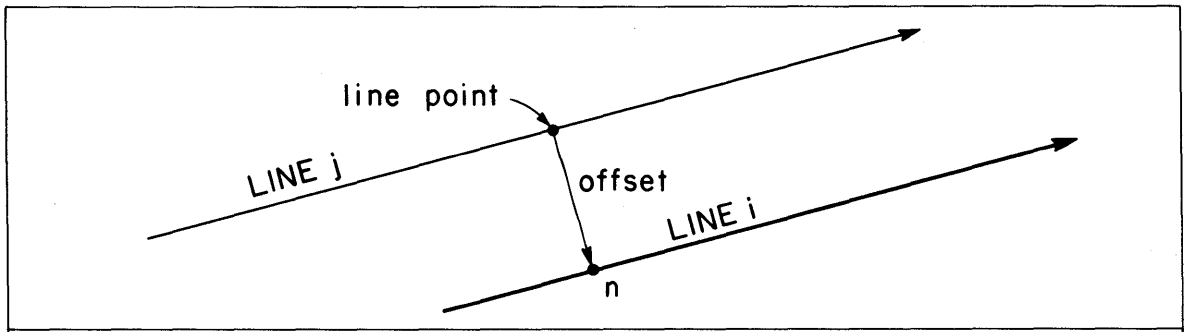


STORE LINE i thru pa at direction | STORE LINE i thru pa TOWARD pb

Compute and Store Parallel Line

STORE LINE i thru n PARALLEL to LINE j, offset, print
STORE LINE i thru n PARALLEL to line, offset, print

Point n is first located at the offset distance from the stored point on the stored line j and the coordinates of n are stored in the Point Table. Line i will have the same direction as line j. If the offset data item is omitted, the two lines will coincide, an offset distance of zero being used. This command stores both Line i and Point n. In the second and more general form, the equivalent of line j is determined from the line data item.

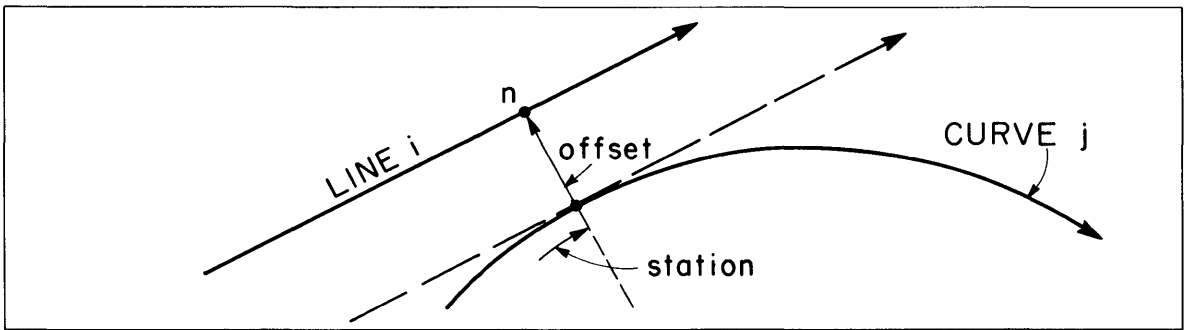


STORE LINE i thru n PARALLEL to LINE j, offset

Compute and Store Tangent Line

STORE LINE i thru n TANGENT to CURVE j at station, offset, print

Point n is first located on stored curve j at the value of station (if offset is specified, n is actually located on the radial to the curve at the offset distance) and the coordinates of n are stored in the Point Table. The forward direction of the tangent to the curve at the value of station is computed for the direction of the line. This command stores both Line i and Point n.



STORE LINE i thru n TANGENT to CURVE j at station, offset

Print Stored Lines

PRINT LINES i, j, k . . . .
PRINT LINES i TO j, (k TO 1), ...
PRINT ALL LINES

The first two forms can be mixed, such as in the following example:

PRINT LINES 14, 8, 50 TO 60, 35, 100 TO 200, 3, 5, 20 TO 30

For each line, the point number and direction are printed.

Delete Lines

To delete lines from the Line Table, the same forms as Print Lines may be used with the command DELETE replacing the command PRINT.

COURSE COMMANDS

Store Course

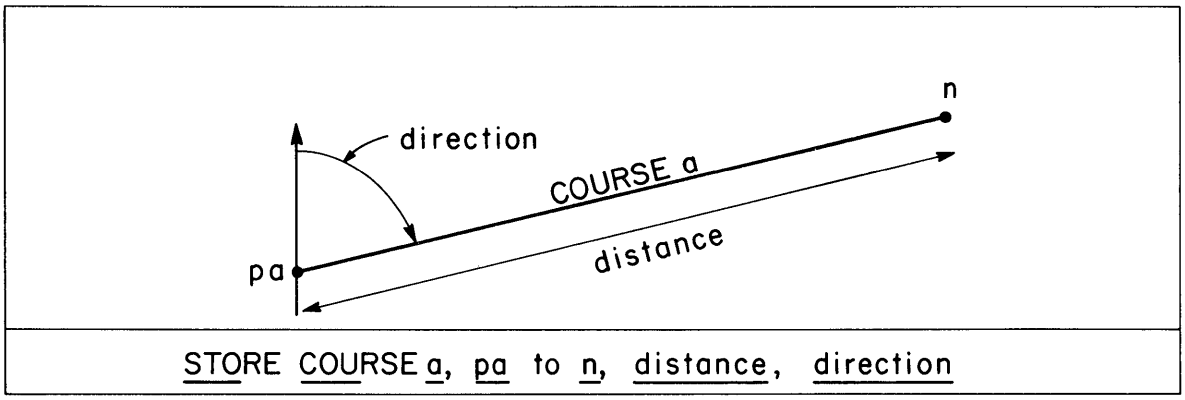
STORE COURSE a pa to pb, print

where a is the name of the course to be stored, pa is the beginning point, and pb is the end point.

Compute and Store Course

STORE COURSE a pa to n, distance, direction, print

The coordinates of n are computed at the values of distance and direction from stored point pa. Point n is stored in the Point Table and then serves as pb in the Store Course command.



Print Stored Courses

PRINT COURSES a, b, c
PRINT ALL COURSES

The value printed is the length and direction of the course.

Compute and Print Courses

PRINT COURSES pa TO pb, (pc TO pd), ...
PRINT COURSES pa TO pb TO pc (TO pd TO pe . . . .)

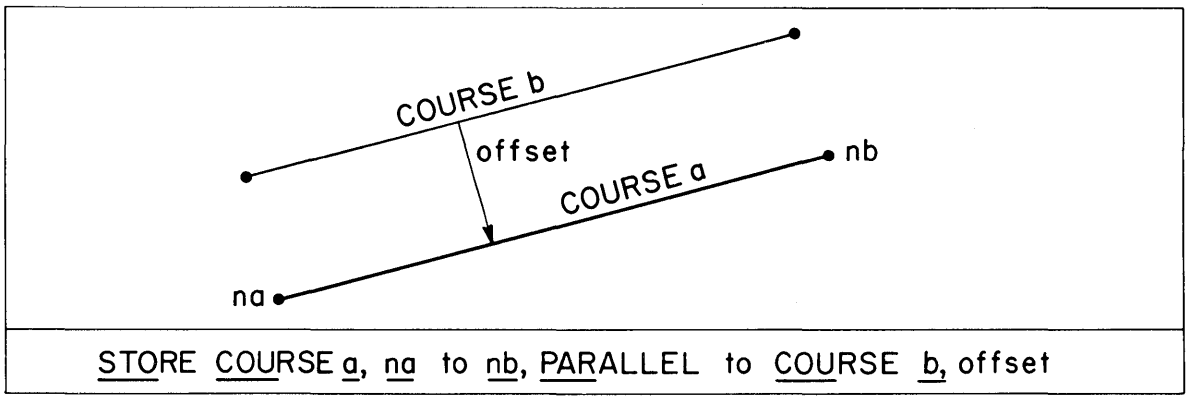
The two forms may be mixed, such as

PRINT COURSES 2 TO 4, 6 TO 8 TO 7, 3 TO 5, PC 2 TO 9 TO PT 3

Compute and Store Parallel Course

STORE COURSE a, na to nb, PARALLEL to COURSE b, offset, print
STORE COURSE a na to nb, PARALLEL to course, print

Points na and nb are computed at the offset distance from the beginning and end points, respectively, of stored course b. Points na and nb are stored in the Point Table and then serve as pa and pb in the Store Course command. If offset is omitted, an offset distance of zero is used, and the two courses will coincide. In the second and more general form, the equivalent of course a and the offset are determined from any of the forms of the course data item.



Delete Stored Courses

DELETE COURSES a, b, c . . . .
DELETE ALL COURSES

Transpose Stored Courses

TRANSPOSE COURSES a, b, c

The order of the beginning point and the end point for course a is transposed in the Course Table; that is, the beginning point becomes the end point, and vice versa. The Station Course command should usually follow this command to provide stationing in the new forward sense.

Station Stored Courses

STATION COURSE a station

See Station command for detailed specifications.

CURVE COMMANDS

Store Circular Curve

STORE CURVE i, reference, element, a/tangent, c/station, print

where i is the identification number of the curve to be stored in the Curve Table and the data items are as described below:

reference: allowable forms - one of the following
- PB at pa, DB direction, TL distance
- PB at pa, DB direction, TTL distance
- PC at pa, DB direction
- PI at pa, DB direction
- PB at pa, PI at pb

element: allowable forms - one of the following

RADIUS	distance
DEGREE	angle
LENGTH	distance
TANGENT	distance
LCHORD	distance
EXTERNAL	distance

CC at pa

a/tangent: allowable forms - one of the following
- DA direction
- p/m DEFLECTION angle
- p/m DELTA angle
- PA at pa (only with 2nd, 4th, and 5th form of reference)
- p/m s/element (only if element is RADIUS or DEGREE)

where s/element is one of the following

LENGTH	distance
TANGENT	distance
LCHORD	distance
EXTERNAL	distance

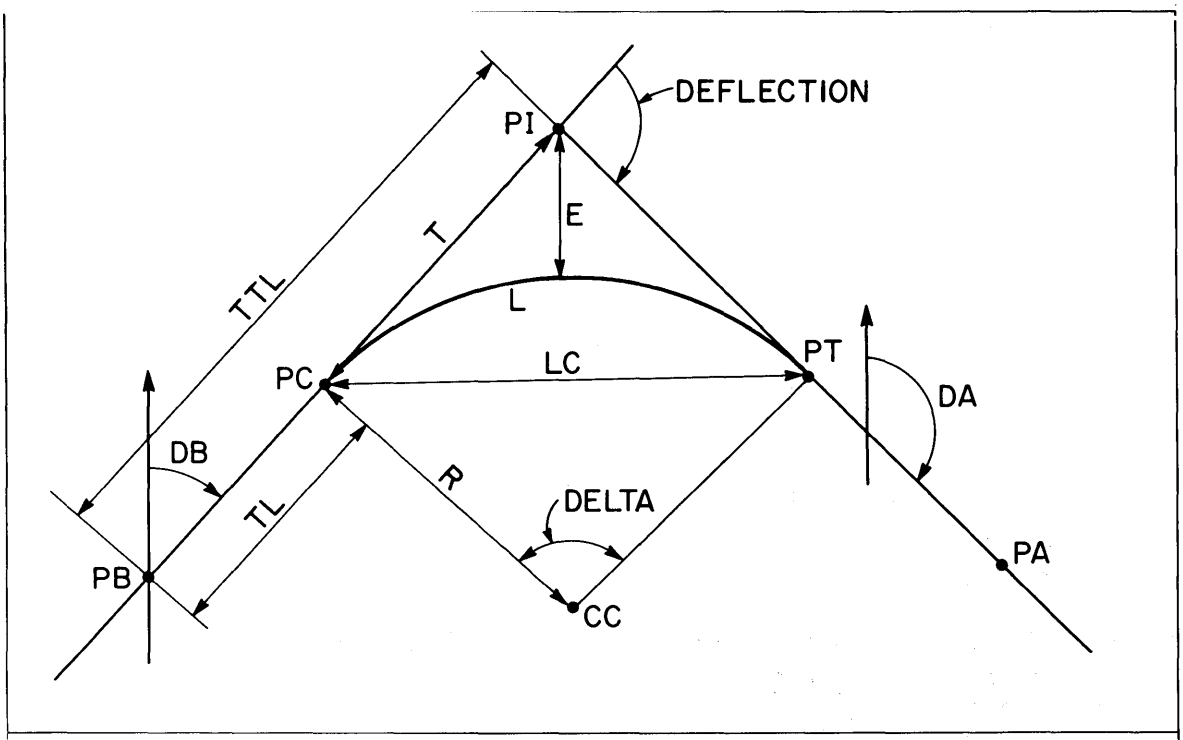
c/station (optional)

STATION of label station

where label is any curve point label except CC. label may also be PB.

If c/station is omitted, PC (or TS) is assumed to have station value of 0+00.

If reference is of the second or fifth form, then element may be TL distance.



Circular CURVE Nomenclature and Data Items

Example Curve Commands

STORE CURVE 1, PB AT 5, DB N 45 E, TL 125.16, R 2000, DA S 35 E
STORE CURVE 2, PC AT 7, DB 4 TO 6, DEG 3, PLUS DEF 78 15
STORE CURVE 3, PI AT 8, DB 270 15, R 3000, MINUS L 1000
STORE CURVE 4, PB AT 4, DB 45 0, CC AT CC 2, PA AT 7
STORE CURVE 5, PB AT PT 3, DB PI 3 TO PT 3, TL 0., T 1000, DA 45
STORE CURVE 6, PC AT 5, DB 'A', DEG 'B', P TAN 4 TO 8
STORE CURVE 7, PB 10, PI 12, R 3 TO 5, PA 20

STORE CURVE 8, PB 2, DB 45, TL 100., SLB 300, R 3000, SLA 200, DA 135
STORE CURVE 9, PI 4, DB 25, SLB 400, DEG 5, SAA 10, P DEF 60
STORE CURVE 10, TS 3, DB 'Z', SLB 500, R 5000, DA 178 56 22
STORE CURVE 12, PB 5, PI 6, DEG 3 30, SL 400, PA AT 44

MODIFY CURVE 2, DEG 2 30
MODIFY CURVE 3, R 3500
MODIFY CURVE 9, SLB 300

STORE CURVE 20, CONCENTRIC WITH CURVE 4, OFFSET P 50
STORE CURVE 22, CONCENTRIC WITH CURVE 6, OFF M 3 TO 5
STORE CURVE 24, PARALLEL TO CURVE 2, OFFSET M 24.
STORE CURVE 26, PARALLEL TO CURVE 24, OFF P 'X3'

PRINT CURVES 1 TO 15, 24, 50 TO 60, 98, 127

Store Circular Curve with Transition Spirals

STORE CURVE i, reference, b/spiral, element, a/spiral, a/tangent, c/station, print

where i is the identification number of the curve to be stored in the Curve Table and the data items are as described below:

reference
Allowable forms same as for store circular curve except that the third form becomes
TS at pa, DB direction

b/spiral (back spiral - spiral from TS) - allowable forms
SLB distance
SAB angle

element (of circular curve segment) - allowable forms
RADIUS distance
DEGREE angle
If a/tangent gives DELTA, then element may also be
LENGTH distance
TANGENT distance
LCHORD distance
EXTERNAL distance

a/spiral (ahead spiral - spiral to ST) - allowable forms
SLA distance
SAA angle

a/tangent: allowable forms - one of the following
DA direction
p/m DEFLECTION angle (deflection angle of total curve)
p/m DELTA angle (delta angle of circular curve)
PA at pa (only with 2nd, 4th, and 5th form of reference)

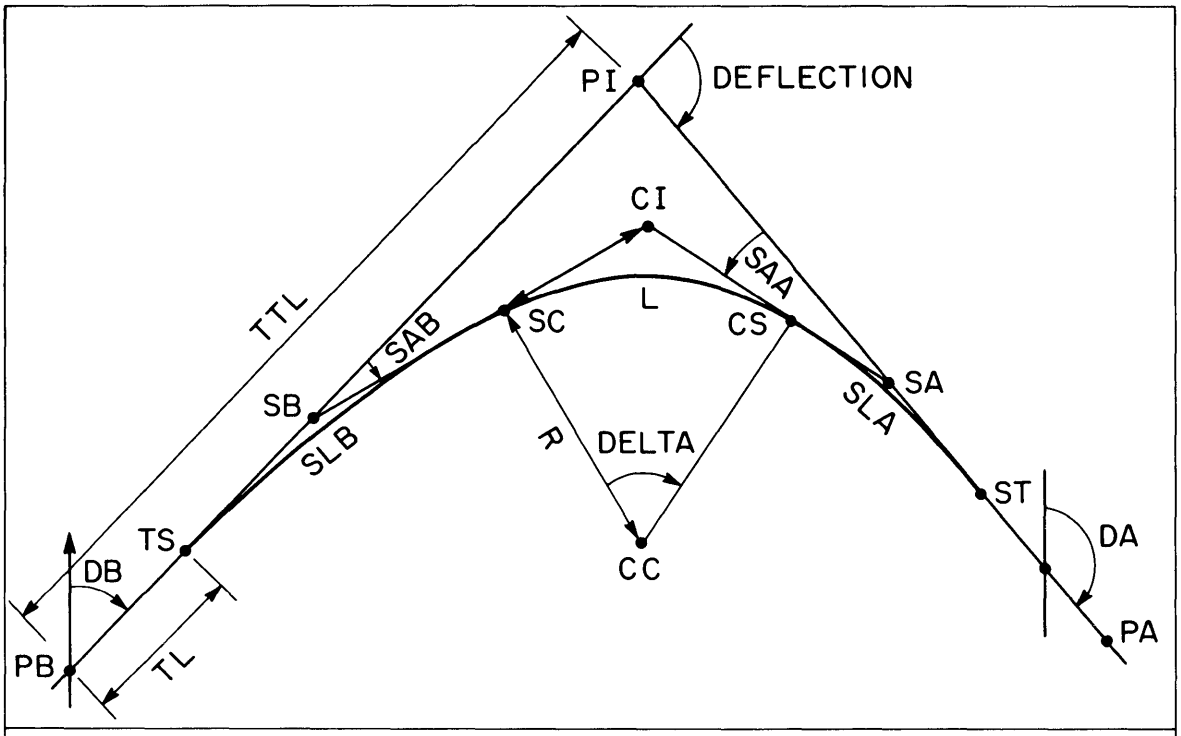
c/station (optional)
Same forms as for Store Circular Curve.

If the curve is to have equal length transition spirals at both ends, then either b/spiral or a/spiral or both may take one of the following forms:

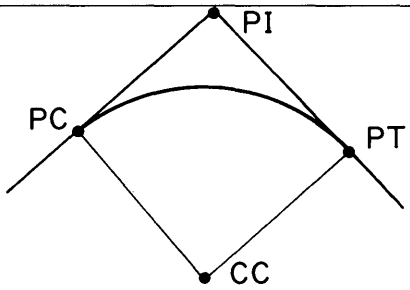
SL distance
SA angle

The same values will be used for both spirals, so in this case one or the other of the data items may be omitted.

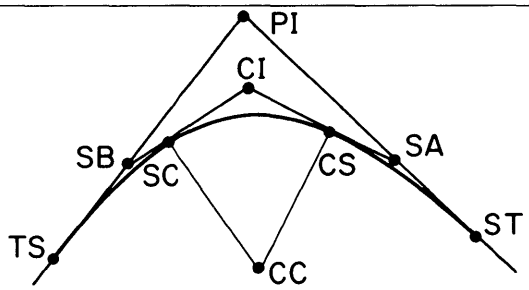
If the curve is to have a transition curve on only one end, the other spiral may be given as a zero length spiral (SLB 0. or SLA 0.) or may be omitted.



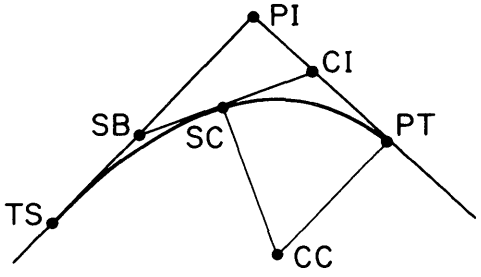
Spiraled CURVE Nomenclature and Data Items



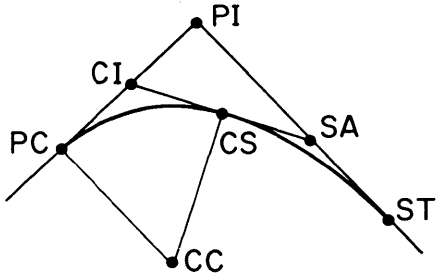
Type C-Circular Curve



Type SCS-Spirals at Both Ends



Type SC-Spiral at Beginning



Type CS-Spiral at End

CURVE TYPES and CURVE POINT LABELS

Modify Stored Curve

MODIFY CURVE i, b/spiral, element, a/spiral, c/station, print

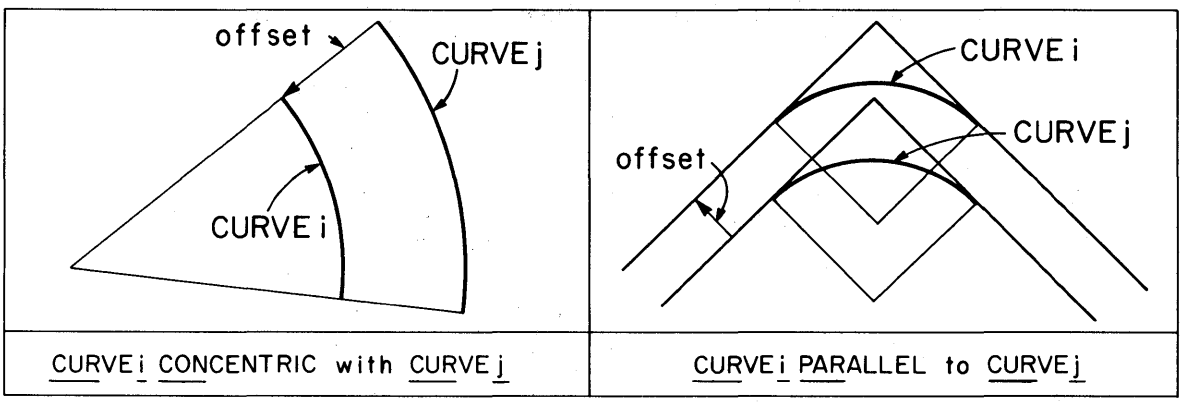
Stored curve i is recomputed and redefined using given new values for the spirals and/or element data items. The command is the equivalent of a Store Curve command except that reference and a/tangent are not given because the stored values which define them are held. The location of the PI and the directions of the back and ahead tangents (DB and DA) are held fixed.

If the spiral data items are not given, the stored values for spiral lengths are held fixed. If the element data item is not given, the stored value for circular curve radius is held fixed. If c/station data item is not given, the stored value for the PI is held fixed in restationing the curve.

Store Concentric Curve

STORE CURVE i, CONCENTRIC with CURVE j, offset, c/station, print

A new curve i is generated which has the same center and tangent directions as stored curve j. The tangents are shifted by the value of offset. This command does not apply to spiral curves.



Store Parallel Curve

STORE CURVE i, PARALLEL to CURVE j, offset, c/station, print

A new curve i is generated which has the same curve element values and tangent directions as stored curve j. The tangents are shifted by the value of offset.

Restation Stored Curve

STATION CURVE i, label station

where label is any curve point label except CC. The stored curve i is restationed with respect to label which is given by the value of station.

Print Stored Curves

PRINT CURVES i, j, k . . . .
PRINT CURVES i TO j, (k TO l), ...
PRINT ALL CURVES

The first two forms can be mixed. The curve point values (coordinates and station) and the curve element values (radius, length, and so forth) are printed.

Delete Stored Curves

To delete curves from the Curve Table, the same forms as Print Curves may be used with the command DELETE replacing the command PRINT.

Transpose Stored Curve

TRANSPOSE CURVE i

Curve i is transposed; that is, the old PT becomes the new PC. A clockwise curve is converted to a counterclockwise curve, and vice versa. The values of the curve elements are unchanged and the position of the curve in space is unchanged (PI and CC unchanged).

A stored curve has a forward sense (PC toward PT) and curve direction (clockwise or counterclockwise) which are defined at the time the curve is stored. Situations can develop where the Engineer may want to reverse the forward sense and curve direction of a stored curve in order to include it in a chain list. The Transpose Curve command facilitates coping with such situations.

Alignment Curves

The Store Curve commands provide for the definition and storage of independent curves. The more common way of defining and storing curves will be via the Curve subcommands of the Alignment command, the procedure for which is defined in the Alignment specifications.

CHAIN COMMANDS

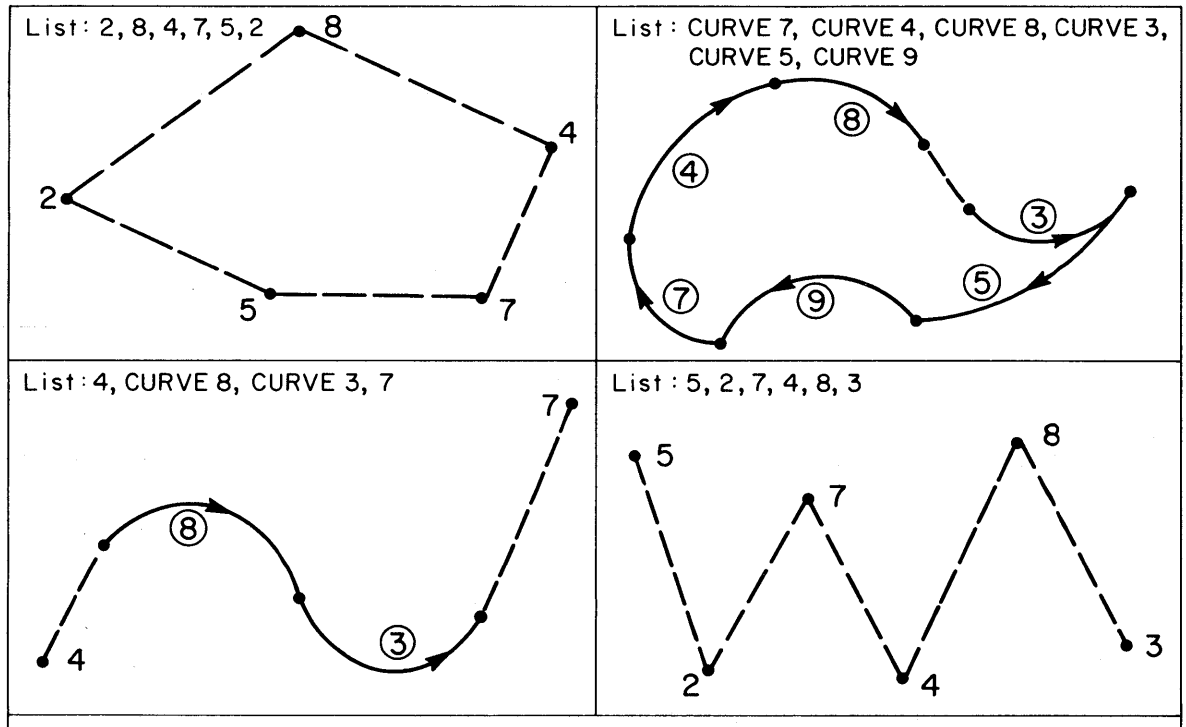
In the command specifications which follow, wherever the word CHAIN is used, any one of the following words may be substituted: TRAVERSE, ALIGNMENT, PARCEL, BASELINE, CENTERLINE, LIST.

Store Chain

STORE CHAIN a, list

where a is the name of the chain to be stored, and list is the appropriate list of stored objects which define the chain. Objects which may be included in the list include:

Objects	Form
Integer Points	(POINT) pi
Curve Points	label i
Courses	COURSE a
Curves	CURVE i
Chains	CHAIN a
Gaps	GAP



EXAMPLES OF CHAIN OBJECTS AND LISTS

Describe Stored Chain

DESCRIBE CHAIN a

A report is printed which describes the detailed geometry of the chain. The report includes the associated text, coordinates of all points, the lengths and directions of all sides, and the standard data for all curves. If the chain is a closed figure (last point in the list is same as first point in list), the area is also printed.

Print Stored Chain List

PRINT CHAINS a, b, c . . . .
PRINT ALL CHAINS

The list stored in the Chain Table for each chain is printed.

Delete Stored Chains

DELETE CHAINS a, b, c . . . .
DELETE ALL CHAINS

The indicated chains are deleted from the Chain Table.

Transpose Stored Chains

TRANSPOSE CHAIN a

The order of all objects in the list for chain a is transposed in the Chain Table; that is, the last object becomes the first, next to last the second, and so forth. Each curve in the list is also transposed (see Transpose Curve command). The Station Chain command should usually follow this command to provide continuous stationing in the new forward sense.

A chain has a forward sense (the forward order of the list) which is defined at the time the chain is stored. Situations can develop where the user may want to reverse or transpose the order. For example, the user may wish to define an alignment with a list that includes already stored chains which are in opposite order to the forward sense of the alignment. The Transpose Chain command facilitates coping with such situations.

Station Stored Chains

STATION CHAIN a (FROM pa) station (AHEAD to pb), (BACK to pc)

See Station command for detailed specifications.

Chain Area

AREA list

where list is the same as for Store Chain; that is, a list of objects which defines the chain which bounds the closed figure for which the area is to be computed (i.e., list defines the perimeter of the figure). The beginning point of the first object in the list should coincide (same point or same horizontal coordinates) with the end point of the last object in the list to define a closed figure. If they do not coincide, a closing link or special connector is generated to close the figure, and the Engineer is notified via a message.

The perimeter or boundary of the figure may include curves, segments of which will be automatically added or subtracted to give the correct net area enclosed by the curvilinear boundary. The area of each curve segment is printed as it is computed. If a curve has spirals, separate segment areas are computed for each spiral and the circular curve, with subchords from the TS to SC to CS to ST.

The list should define a single figure; that is, the boundary should not intersect itself. If it does cross itself, resulting in multiple closed figures, the area computed will not have meaning.

To compute and print the area of a single stored chain, the command would be

AREA CHAIN a

and to compute and print the segment areas of a single stored curve, the command would be

AREA CURVE i

Example Chain Commands

```
STORE PARCEL 'A' 4, 3, CURVE 8, 7, 5, 4
STORE ALIGNMENT 'RAMP-J' 12, CURVE 5, CURVE 10, 27
STORE TRAVERSE 'BOUND' 16, 20, 4, 10, 15, 94, 25
STORE ALIGN 'B', ALIGN 'A', ALIGN 'RAMP-J', ALI 'C'
STORE BASELINE 'B24-X', TRA '2', 18, TRA 'X12'
DESCRIBE PARCEL 'A'
DESCRIBE CHAINS 'RAMP-J', 'BOUND', 'B24-X'
STATION ALIGN 'B' FROM PC 12 STA 5+00, AHEAD TO PT 4
STATION ALIGN 'RAMP-J' 0+00
AREA PARCEL 'A'
AREA 2, 4, 6, 8, 5, CURVE 10, 2
```

TEXT COMMANDS

Store Text

STORE TEXT a, n

where a is the name of the set of text and n is the number of lines (cards) which follow the command and which contain the text. When the command is executed, the next n cards are read and stored in the Text Table as Text a. The text can contain any legal characters.

Text Output

Whenever the command DESCRIBE CHAIN a is executed, the Text Table is checked. If there is a stored text with the same name, that set of text is printed before the regular output report for the Describe command. The text printout is a card image of the text which is stored.

Stored text may also be printed with the commands:

PRINT TEXTS a, b, c . . .
PRINT ALL TEXTS

and may be deleted with the commands:

DELETE TEXTS a, b, c
DELETE ALL TEXTS

Example Store Text

```
STORE TEXT 'LOT/18', 7
    PARCEL LOT/18 IS LOCATED IN SUNSHINE SUBDIVISION,
    TOWN OF LEXINGTON, MIDDLESEX COUNTY, MASSACHUSETTS,
    IN THE NORTHWEST QUADRANT OF THE INTERSECTION
    OF OLD COLONY ROAD AND MINUTEMAN LANE.
    OWNED BY SMITH REAL ESTATE CORPORATION.
    OCCUPIED BY SERVICE STATION AND PARKING LOT.
    ASSESSED VALUATION $376,500.
```

STATION COMMAND

Station Point

STATION pj station

Stored integer point pj is given the value of station.

Station Points on Curve

STATION ON CURVE i, (POINTS) pi, pj, pk . . . .

The stations of the stored integer points in the list are computed on stored curve i and are stored in the Point Table. If a point is not on the curve (±. 1), the station is computed for the radial projection of the point onto the curve or nearest tangent to the curve, a message being printed. Stationing is always computed with respect to the station of the beginning point of the curve segment except when the point is projected onto the ahead tangent. Then station is based on the station of the end (PT or ST) point of the curve.

Restation Stored Curve

STATION CURVE i, label station

where label is any curve point label except CC. The stored curve i is restationed with respect to label which is given the value of station.

Station Points on Course

STATION ON COURSE a, (POINTS) pi, pj, pk . . . .

The stations of the stored integer points in the list are computed on stored course a and are stored in the Point Table. If a point is not on the course (±. 1), the station is computed for the projection of the point onto the course or course extension, a message being printed. Stationing is always computed with respect to the beginning point of the course, and stationing is assumed to be increasing in the direction toward the end point, with one exception. If the station of the end point is stored and is compatible with the station of the beginning point (absolute value of difference in station equals length of course) and indicates descending stationing, then the point will be correctly stationed in the descending stationing.

Station or Restation Stored Course

STATION COURSE a station

The stored course a is stationed with respect to the beginning point. If both points are integer points, the beginning point is given the value of station and the end point is given the value of station plus the length of the course. If the end point is a curve point, its station value is left unchanged and the beginning point is given the value of station. If the beginning point is a curve point, its station value is left unchanged and the end point is given the value of station plus the length of the course. If both points are curve points, no stationing will be performed and the user will be so notified.

Station or Restation Stored Chain

STATION CHAIN a (FROM pa) station (AHEAD to pb) (BACK to pc)

Stored chain a is stationed or restationed continuously, starting at point pa, which is assigned the value of station. Stationing is computed forward to point pb and is backed up to point pc. pa, pb, and pc must be points in the chain list for chain a (pa may be a PC or TS curve point; pb may be a PT or ST curve point; pc may be a PC or TS curve point). Every point, course, curve, and chain in the chain list from pb to pc is stationed or restationed, the station values being stored in the data tables.

If pa is not given, it is taken to be the point at the beginning of the chain. If pb is not given, it is taken to be the point at the end of the chain. If pc is not given, it is taken to be the same point as pa (i.e., no back stationing). For example, if chain 'A' is stored as

STORE ALIGNMENT 'A' 2, 8, CURVE 5, CURVE 7, 4, 6

the command

STATION ALIGN 'A' STA 5+00

will store 5+00 as the station of Point 2 and will station 'A' continuously from Point 2 to Point 6.

The command

STATION ALIGN 'A' FROM PC 5, STA 10+00, AHEAD TO 4,
BACK TO 8

will store 10+00 as the station of the PC of Curve 5, will back up the stationing to Point 8, and will compute and store stationing ahead to Point 4.

PROFILE COMMANDS

Profile Command

The command calls a general purpose profile subprogram which processes a set of building block subcommands which define and store a vertical alignment profile in the Profile Table. The general form of the command and the order of the subcommands is as follows:

PROFILE a
VPI subcommands (three or more)
END of PROFILE, print

where a is the name (1 to 8 characters enclosed in single quotes) of the profile and the VPI subcommands are as described in the section which follows. If the optional print modifier is given with the required END command, a Print Profile command is executed.

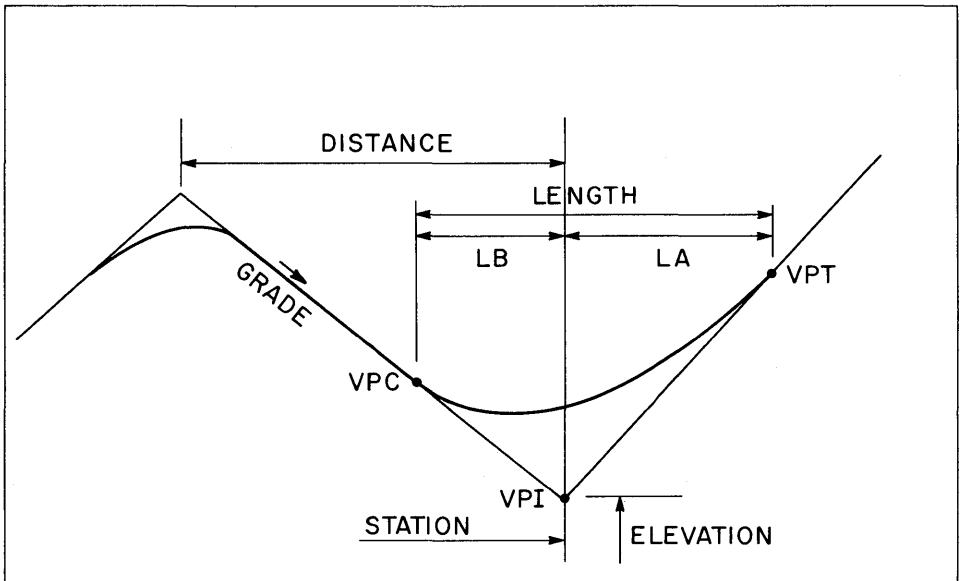
VPI Subcommand

VPI at pj, station, elevation, length

where pj is the identification number of the integer point at the VPI. If station is not given, it is understood that the station of pj is already stored in the Point Table, and the stored value is used in the profile computations. If elevation is not given, it is understood that the elevation of pj is already stored in the Point Table, and the stored value is used in the profile computations. If length is not given, it means that no vertical curve is to be generated at the VPI. The first (initial point) and last (terminal point) VPI subcommands should not include the length data item. When the data items are given, they take the following forms:

Data Item	Allowable Forms
station	STATION v where v is the station of pj
	DISTANCE v the station of pj is computed by adding v to the station of the previous VPI

Data Item	Allowable Forms
elevation	ELEVATION v or Z v where v is the elevation of pj
	ELEVATION at station on (PROFILE) b where b is the name of a stored profile. The elevation of pj will be computed at the value of station on profile b.
	GRADE p/m v where v is the grade in decimal percent of the tangent from the previous VPI to pj (PLUS is assumed if p/m is not given). The elevation of pj is computed using the grade value and the distance value.
	GRADE at station on (PROFILE) c where c is the name of a stored profile. The grade will be computed at the value of station on profile c. This grade value will then be used to compute the elevation of pj as in the previous form.
	The AT forms are provided for use in computing ramp profiles and other conditions where the profile being computed is constrained by another (stored) profile.
length	LENGTH v where v is the length of the symmetric vertical curve at the VPI.
	LB vb, LA va where vb is the length back and va is the length ahead of an asymmetric vertical curve at the VPI.



PROFILE and VPI Nomenclature and Data Items

If the given length causes the vertical curve to overlap the previous vertical curve, an asymmetric vertical curve length back which will eliminate the overlap is used and the Engineer is so notified.

Unknown Conditions in VPI Subcommand

If in the elevation data item the elevation and the grade are given; that is, the command is of the form:

VPI at pj station, elevation, grade, length

where elevation is either of the first two forms of the elevation data item and grade is either of the second two forms of the elevation data item, the VPI subcommand of this form may be preceded by one VPI subcommand of the following form:

VPI at n station, GRADE ? length

or

VPI at n STATION ?, grade, length

The unknown grade or station will be back computed from the VPI with station, elevation, and grade. The two VPI's can be separated by any number of VPI subcommands of the form:

VPI at n, station, grade, length

There can be more than one unknown condition in a profile so long as each subcommand with an unknown (?) is followed by a subcommand with both elevation and grade values in the elevation data item.

Note: The word UNKNOWN may be used in place of the symbol ? and should be used when the problem is to be run at an installation which does not have the ? character on its printer chain.

Shift VPI

Since the VPI is defined by an integer point stored in the Point Table, it may be shifted by changing the station and/or elevation of that point by use of the Store Point command or any other command which changes these entries in the Point Table. Since the Profile is a relative object and is dependent on the entries in the Point Table, the Engineer must be careful that he does not unintentionally shift a Profile by changing a point station or elevation.

Change or Store Vertical Curve Length

MODIFY VC at (VPI) pj (ON) (PROFILE) a, length

The Profile Table entry for the vertical curve length of the vertical curve at pj on stored Profile a is changed to the value of length, where length is as described for the VPI subcommand. The new value of length replaces the old value and will be used in generating the profile when it is referred to in subsequent commands.

Print Stored Profile

PRINT PROFILES a, b, c . . . . .
PRINT ALL PROFILES

A report is printed which describes the detailed geometry of each profile. The report includes such standard information as the station and elevation of the VPC, VPI, and VPT of each vertical curve, its length, grade back and ahead, high or low point location, and so forth.

Print Elevations on Stored Profile

PRINT ELEVATIONS ON (PROFILE) a list

where each item in list can be a numerical station value or an item of the form:

EVEN v (STATIONS) va TO vb

Example:

```
PRINT ELEV ON PROFILE 'P2', 6+25, 7+19, EVEN 50 10+00 TO 20+00,
    15+76
```

Elevations will not be printed beyond the limits of the stored profile. If station values in the list are given in order of increasing magnitude, the speed of processing is greatly increased.

Delete Stored Profiles

DELETE PROFILES a, b, c . . .
DELETE ALL PROFILES

The indicated profiles are deleted from the Profile Table.

Constrained Vertical Curves

In the VPI subcommand (page 36-37), the length data item may take the following special forms to fit a vertical curve to given constraints:

THRU pk
THRU STATION v, ELEVATION v
CLEAR pk BY P/M v
CLEAR STATION v, ELEVATION v BY P/M v
FIXED (GRADE) v AT (STATION) v

The length of a symmetrical vertical curve is computed internally to fit the specified constraint. If LENGTH v is given and one of the above constraints, then an asymmetrical vertical curve of total length v is computed internally which fits the constraint.

Example Profile Commands

```
PROFILE 'MAIN'
VPI AT 27 S 0+00 Z 1000
VPI 30     S 10+00 G 2.25 L 700
VPI AT 24 LB 800. LA 600.
VPI 43     S 56+17 Z 1256.47 L 1200
VPI 15 STA ?, G M 3.5, L 1000
VPI 25, S 179+25, G 1.5, L 800
VPI 35, S 214+00, Z 1123., G M 2., L 1500
VPI 7, DIST 2000., G AT 25+00 ON 'RAMP', L 1200
VPI 8, S 342+00, E 15+00 ON 'AX', LB 700 LA 800
VPI AT 45 S 400+00 G ?, L 2000
VPI AT 50
END OF PROFILE, PRINT

PRINT ELEV ON 'MAIN', EVEN 20, 80+00 TO 300+00
STORE VPI 15 ON PROFILE 'MAIN', L 800
STORE 8 S 345+00
PRINT PROFILE 'MAIN'
```

LOCATE COMMAND

Introduction

The Locate command provides a variety of ways to locate (compute and store the coordinates) of a point with respect to one or more stored objects. The basic cases and forms of the command are as follows:

1) Locate From

To locate a point at a given distance and direction from a stored point. The distances and directions can be given in a variety of ways. A series of Locate From commands is the equivalent of an open traverse. The basic form is

LOCATE n from pa distance, direction, offset, print

2) Locate On

To locate a point on a stored or defined object. The object may be a line, course, or curve. The location on the object can be given as a station or as a distance from a stored point. The basic form is

LOCATE n ON object p/m distance from pa, offset, print

3) Locate Intersection

To locate a point at the intersection of two stored or defined objects. The two objects can be any combination of lines, courses, and curves, with or without offsets. The basic form is

LOCATE n, INTERSECT object (WITH) object, (NEAR pa), print

4) Locate Projection

To locate a point as the projection of a stored point onto a stored or defined object. The object may be a line, course, or curve, with or without an offset. The basic form is

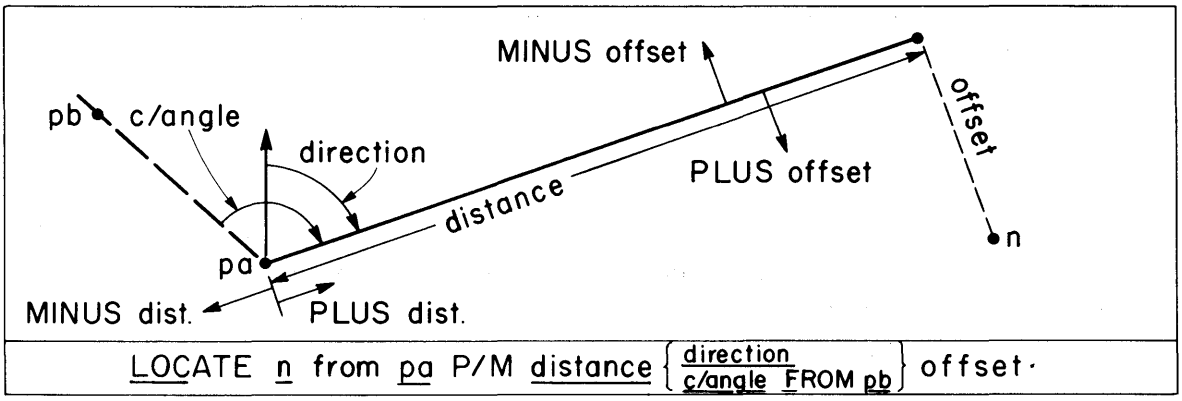
LOCATE n, PROJECT pa on object, print

Special forms are also provided for locating points by foresection and by distance and angle ties. The specifications for all forms are described in the sections which follow.

Locate From Point

LOCATE n from pa p/m distance, direction, offset, print
LOCATE n from pa p/m distance, p/m angle FROM pb, offset, print

Point n is located at the value of distance from pa at the value of direction (in the second form, the value of direction is computed as the direction from pa to pb, PLUS or MINUS the value of angle). A MINUS distance causes n to be located in the opposite direction from the value of direction (PLUS is assumed if not given). If offset is given, n is actually located on a line parallel to that defined by pa and direction and at the offset distance.



Example Locate From and On Commands

LOCATE 4 FROM 8 DIST 176.15 AZ 135 15 20
LOC 4 8 176.15 135 15 20
LOCATE 5 FROM 4 7 TO 15 N 75 25 W
LOC 6 5 'X' AZ 4 TO 7 PLUS ANGLE AT 2, 8, 10
LOC 20 4, M 100, P 138 15 FROM 6, OFFSET P 24.

LOCATE 12 ON LINE 5 100. FROM 2
LOCATE 14 ON CURVE 4 50 FROM PC 4
LOCATE 13 ON CURVE 8 AT STA 127+57.16
LOC 18 ON CURVE 8 24. FROM 13 OFF P 20.

Locate On Object

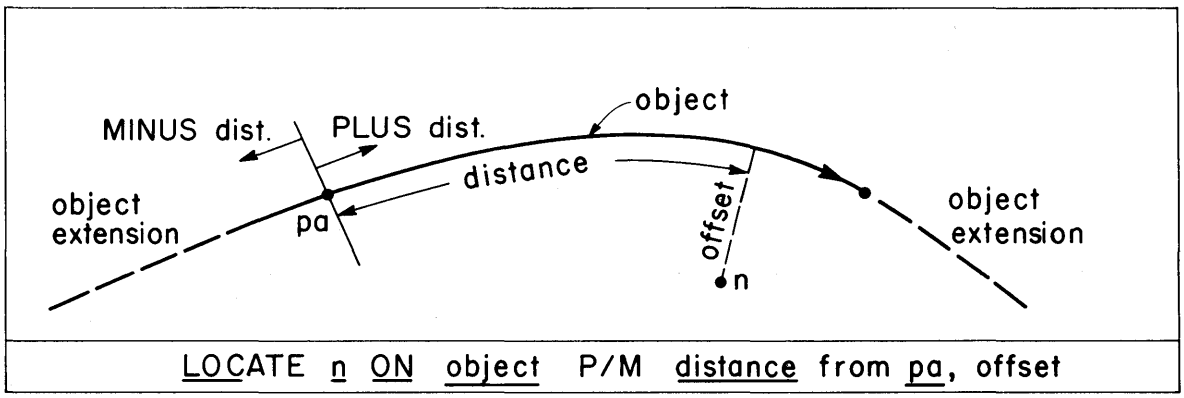
LOCATE n ON object p/m distance from pa, offset, print
LOCATE n ON object at STATION station, offset, print

The object provides the direction along which distance is measured to locate n. object can take any of the standard allowable forms for line, course, or curve but without the object offset modifier. pa must be a stored point on the object. If it is not on (within ± 0.1), the command is considered to be in error and is not executed. PLUS distance means n is ahead of pa, MINUS distance that n is behind pa. If necessary, the object will be extended to locate n. If offset is given, n is located on the perpendicular or radial to the object at the offset distance. If object is a curve and n is located on the curve (not on an offset), n is stationed on the curve. If object is a course and n is located on the course (not on an offset), n is stationed on the course if the stations of the end points are stored and are compatible.

The second form is processed the same as the first form with the following prior action taken internally:

a) If object is a line, pa is taken to be the line point. If object is a course, pa is taken to be the beginning point of the course. If object is a curve, pa is taken to be the beginning point (PC or TS) of the curve.

b) The value for distance is then computed by subtracting from the given value of station, the stored station for the point taken to be pa.



LOCATE n ON object P/M distance from pa, offset

Locate On Tangent

LOCATE n ON CURVE i, sign, at TANGENT definition, offset, print

Point n is located and stationed on the curve at a point of tangency defined by definition, where definition can take one of the following forms:

direction
: The tangent to the curve at n will have the value of direction.

THRU pb
: A line thru pb will be tangent to the curve at n.

TO CURVE j, sign
: The tangent to Curve i will also be tangent to curve j.

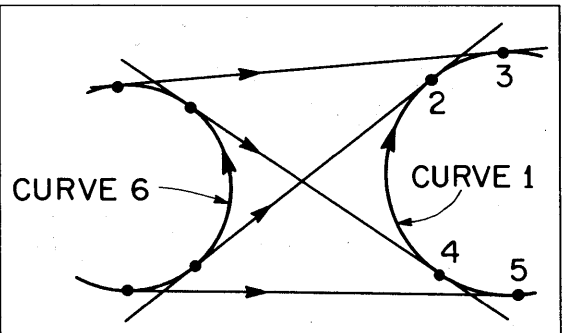
The required data item sign (which can appear in two places in the command) takes one of the following forms:

CL
: Clockwise continuation of the curve beyond n in the same direction as that of the tangent to the curve

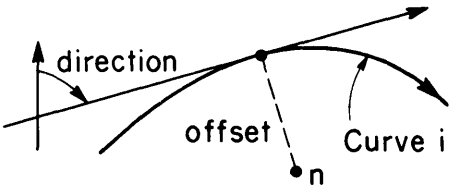
CC
: Counterclockwise continuation

If the offset data item is given, n is actually located on the radial to the curve at the offset distance from the computed point of tangency. If the tangent point cannot be located on the curve, the command is considered to be in error and the point is not located.

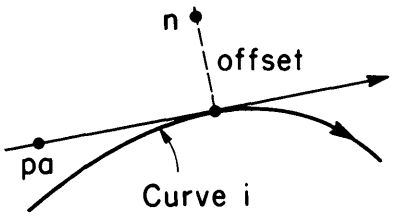
```
LOC 2 ON CURVE 1, CL, TAN TO CURVE 6, CC
LOC 3 ON CURVE 1, CL, TAN TO CURVE 6, CL
LOC 4 ON CURVE 1, CC, TAN TO CURVE 6, CL
LOC 5 ON CURVE 1, CC, TAN TO CURVE 6, CC
```



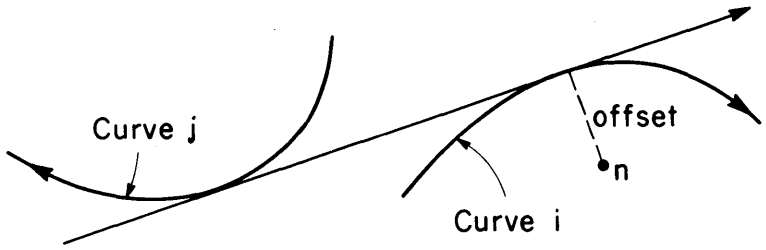
Examples of Use of Sign data item in Locate On Tangent



LOC n ON CURVE i TANGENT direction, offset



LOC n ON CURVE i TANGENT THRU pa, offset



LOCATE n ON CURVE i TANGENT to CURVE j, offset

Locate Intersection

LOCATE n INTERSECT object with object (NEAR pa), print

Point n is located at the intersection of the first object with the second object. Each object can be any of the standard allowable forms for line, course, or curve and either or both objects may have an offset. If offset is given with the object, the intersection is computed on the offset to the specified object.

The optional data item NEAR pa may be used when two or more intersections are possible. The location assigned to n is the intersection nearest to pa. If multiple intersections are possible and the near point is not given, an assumed near point is taken from the first of the two objects as follows: If the first object is a line, pa is taken to be the line point; if a course, the beginning point of the course; if a curve, the beginning point of the curve. One or both objects will be extended if necessary to achieve an intersection.

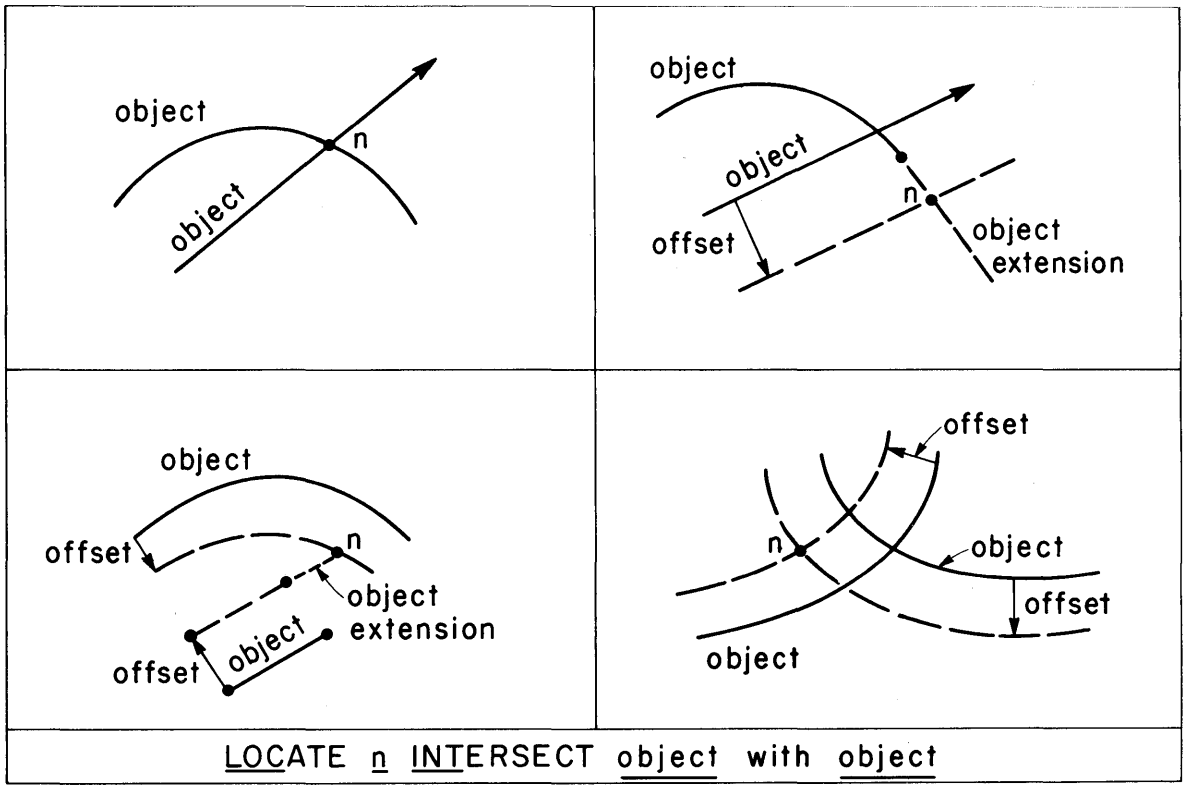
Point n is stationed as follows:

a) If the first object is a curve (without offset), and if the intersection occurs on that curve (not on an extension), n will be stationed on the curve.

b) If the first object is a course (without offset), and if the intersection occurs on that course (not on an extension), and if the stations at the end points are stored and are compatible, n will be stationed on the course.

If neither of the above conditions is fully met, Point n will not be stationed.

When no intersection is possible even if the objects are extended, n is not located and an error message is printed. No intersection cases include lines or courses which are parallel or nearly parallel, curves which are concentric, and lines or courses which are outside of and miss curves. Lines or courses are considered nearly parallel if the acute angle of intersection is less than 0.1 degree. If the acute angle of intersection is less than one degree, a warning message designating a weak solution is printed, but the point is located.



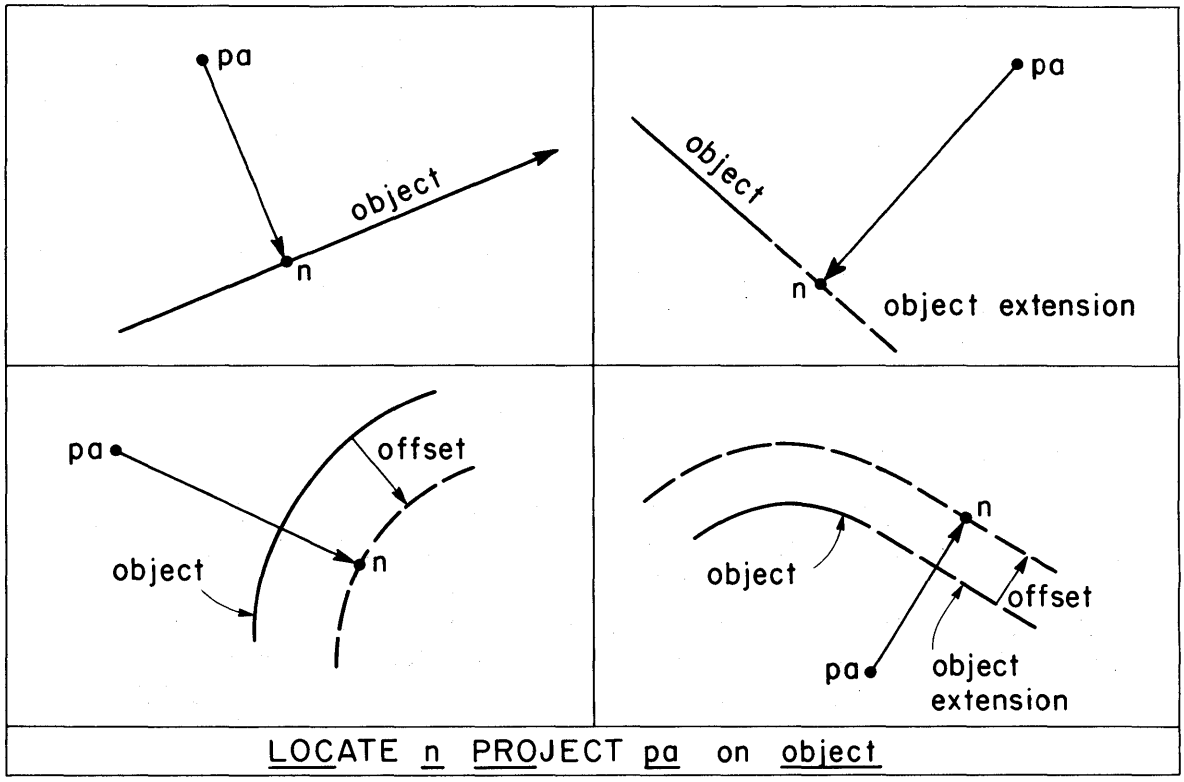
Example Locate Intersect Commands

LOCATE 42 INTERSECT LINE 8 WITH LINE 4, PRINT
LOC 48 INT LINE THRU 3 AT N 15 W WITH COURSE 2 TO 5
LOC 35 INT CURVE 4 WITH LINE 6 NEAR PT 4
LOC 45 INT CURVE 2, OFFSET P 12. WITH CURVE 5, OFF M 10.
LOC 36 INT COURSE 'A8' WITH COURSE PC 5 TO PI 5
LOC 44 INT CURVE 3 WITH LINE THRU 5 AT 6 TO 8, NEAR 14

Locate Projection

LOCATE n PROJECT pa on object, print

Point n is located at the projection of pa onto object. object can be any of the standard allowable forms for line, course, or curve, with or without an offset. If object is a line or course, the projection is perpendicular to the object. If object is a curve, the projection is radial to the curve. If offset is given with the object, n is located on the offset. The object will be extended, if necessary, to achieve the projection. Point n will be stationed on the object if the object is a curve or a course without offset and if not extended.



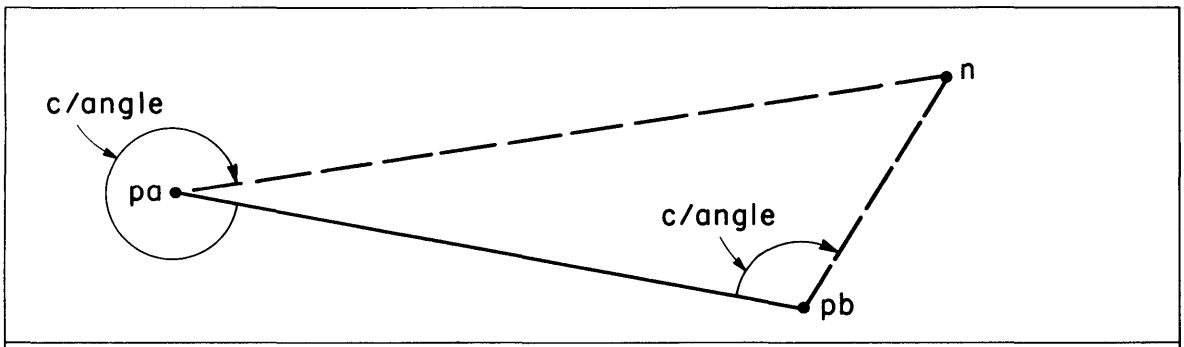
Example Locate Project Commands

LOCATE 55, PROJECT 8 ON CURVE 4
LOC 56, PROJ 10 ON COURSE 2 TO 7, OFFSET P 47.15
LOC 57, PROJ 12 ON LINE THRU 3 AT AZ 5 TO 6 P 90
LOC 58, PROJ 13 ON CURVE 5, OFF M DIST 'X', PRINT
LOC 59, PROJ 14 ON LINE 5, OFF P PI 7 TO PT 14

Locate Foresection

LOCATE n FORESECT pa c/angle AND pb c/angle, print

Point n is located at the intersection of the two lines defined by the points and the computed directions (direction as a data item can be given in place of either or both c/angle data items). The first c/angle is the angle at pa from pb to n. The second c/angle is the angle at pb from pa to n. This is a special case of the intersection of two lines.



LOCATE n FORESECT pa, c/angle AND pb, c/angle

Example Locate Foresect Commands

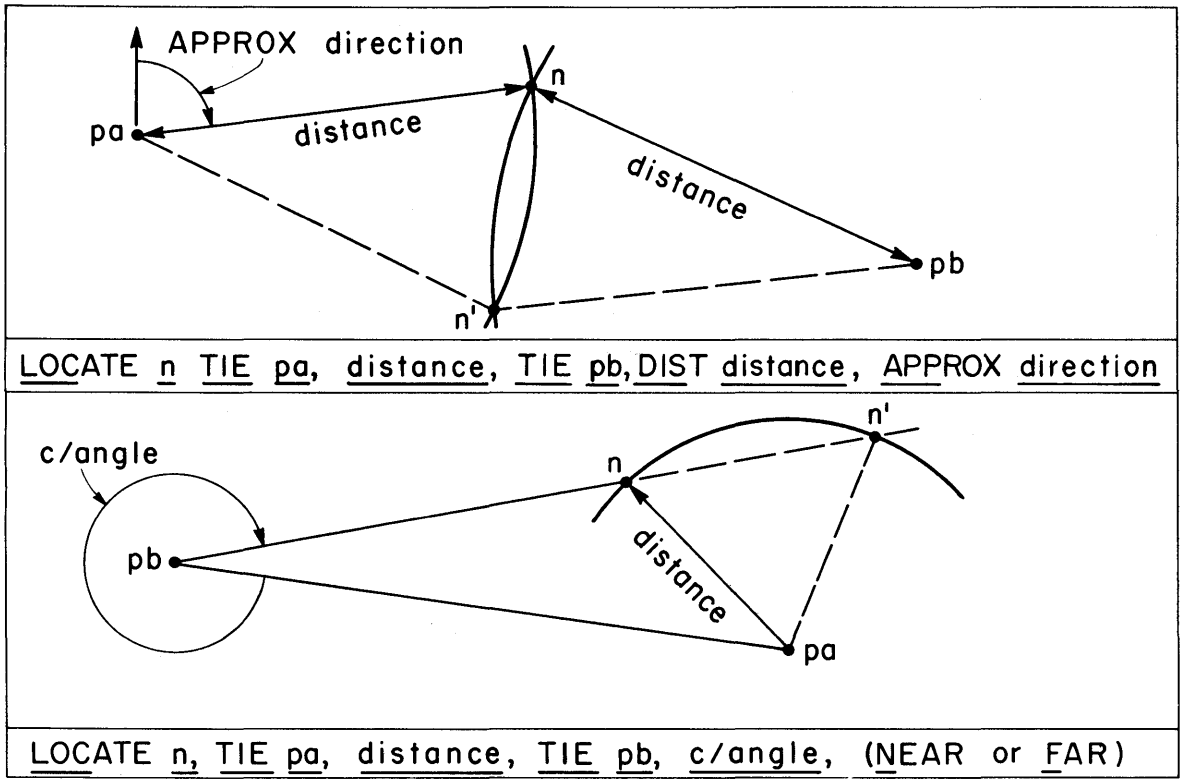
```
LOCATE 70 FORESECT 5, P 305 04, AND 8, P 62 14 59
LOC 72 FOR 3, N 45 E, AND 6, N 37 W
LOC 74 FOR 10, M ANG AT 2 FROM 5 TO 7, AND 12, AZ 'A'
LOC 76 FOR PC 4, P 'A1', AND PI 7, AZ 3 TO 5
```

Locate By Ties

LOCATE n TIE pa distance, TIE pb DISTANCE distance, APPROX direction, print
LOCATE n TIE pa distance, TIE pb c/angle, (NEAR or FAR), print

The first form is the equivalent of intersecting two circles with centers at pa and pb and with radii equal to the values of distance, respectively. Since two intersections are possible, the approximate direction from pa to n is required to determine which intersection to use for n.

The second form is the equivalent of intersecting a circle with center pa and radius distance with a line through pb at a computed direction (direction as a data item can be given in place of c/angle). c/angle is the angle at pb from pa to n. Two solutions are possible. NEAR is given when n is the solution nearest to pa, FAR when it is the farthest solution. NEAR is assumed if neither is given.



Example Locate Ties Commands

LOCATE 4 TIE 2 156.43, TIE 8 DIST 3 TO 5, APPROX 45
LOC 20 TIE PC 5 'X', TIE PT 3 DIST 100, APPROX 4 TO 5
LOC 30 TIE 8 DIST 5 TO 7, TIE 7 PLUS ANG AT 7, 8, 5, NEAR
LOC 40 TIE 5 100 TIE 2 N 35 E, FAR, PRINT

ALIGNMENT COMMAND

Introduction

The Alignment command and associated subcommands can be used to solve a wide variety of horizontal alignment problems, ranging from the simplest to the very complex. Each subcommand defines a point (POT) or curve in the alignment and adds an object to the chain list for the alignment. The alignment is automatically stored in the Chain Table for later retrieval.

Three basic types of alignment problems are handled--Open, Incomplete, and Inverse. An Open Alignment is analogous to an open traverse; that is, each curve or point is independent of those ahead. An Incomplete Alignment is analogous to a closed traverse with missing parts; that is, the alignment must be computed as a whole to determine the unknowns. An Inverse Alignment is analogous to an inverse traverse; that is, the coordinates of each PI and point are known but may be subject to adjustment.

General Form

The general form of the command and the order of the subcommands is as follows:

ALIGNMENT a, type
Initial Point POT Subcommand
Curve and POT Subcommands
Terminal Point POT Subcommand
END of ALIGNMENT, (DESCRIBE)

where a is the name of the alignment. The alignment is stored in the Chain Table. type is one of the following:

OPEN, INCOMPLETE, or INVERSE, round

If DESCRIBE is given, a DESCRIBE CHAIN a command is executed after the processing and execution of all subcommands.

The Initial Point, Curve, POT, and Terminal Point subcommands describe, in order, the points and curves which define the alignment. Detailed subcommand specifications are given with the specifications for each type alignment. General information applicable to all three types is given in the sections which follow.

The END subcommand terminates the processing of the curve and point subcommands. It must be included. The number of curve and point subcommands is limited only by the data table capacities--1,000 curves and 10,000 points.

POT Subcommands - General Information

A POT is a point which is used to define an alignment. It might be any of the following:

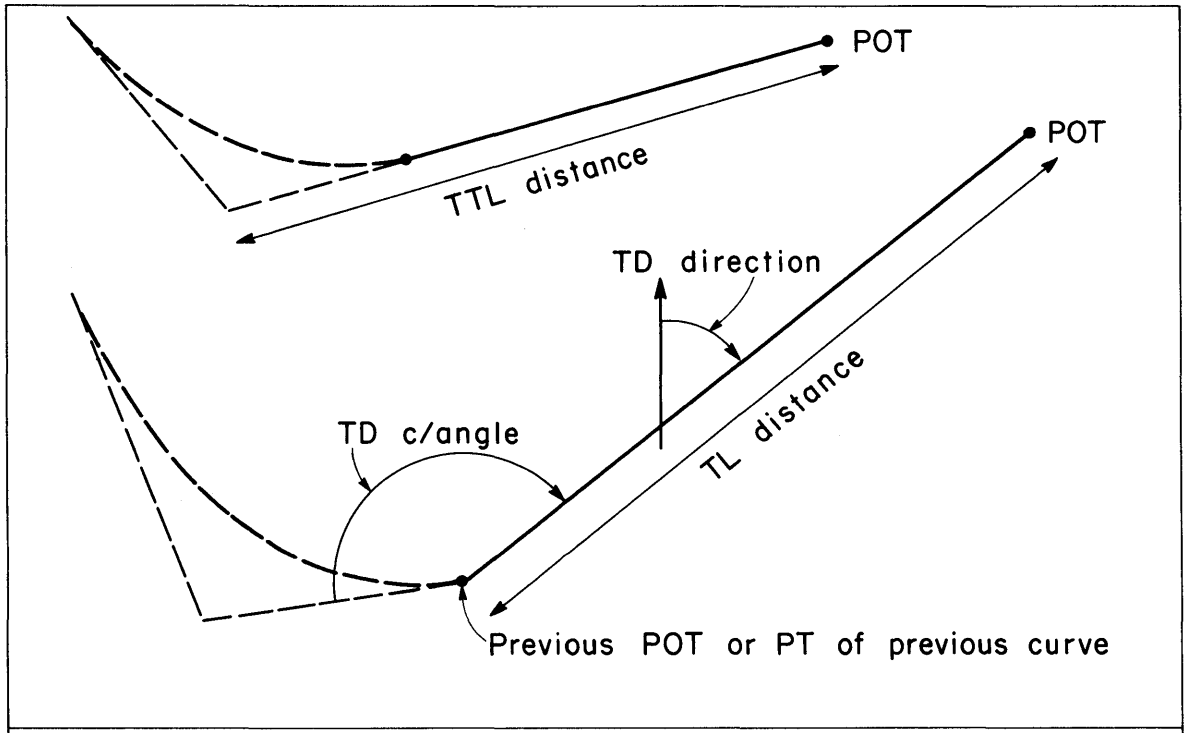
a) The initial or starting point of the alignment

b) An intermediate point in the alignment which may be a point on tangent, or an angle point (equivalent of a PI with no curve)

c) The terminal or ending point of the alignment

POT's are integer points and are stored in the Point Table with their coordinate values and their station on the alignment. Each POT is added to the chain list for the alignment.

The POT subcommand provides the information to determine the coordinates of the point; either directly as an independent object, or indirectly as a dependent object located with respect to the previous curve or point.



Alignment POT Nomenclature and Data Items

As an independent object, the subcommand takes the general form:

POT p/data . . . . . (STATION station)

where p/data takes one of the following forms:

a) n coord

where n is the point number assigned to the POT and coord is the numerical values of the coordinates, such as

POT 4 X 1000. Y 2000. . . . . .
POT 8 N 2500. E 3000. . . . . .

b) n at pa

where n is the point number assigned to the POT and pa is any stored point. n is given the same coordinates as those for pa. Examples

POT 6 AT PC 15 . . . . .
POT 9 AT POINT 3 . . . . .

c) at pj

the point number and the coordinates of stored integer point pj are assigned to the POT. Examples

POT AT 5 . . . . .
POT 2 . . . . .

(Note--pj will be stationed on the alignment, the old station, if any, being lost.)

d) at IP

This form is used, in the case of the Incomplete Alignment, for the Terminal Point POT subcommand when the terminal point is the same point as the initial point.

When the independent form is used for an intermediate POT, the POT will usually be an angle point (a PI without a curve).

As a dependent object, the subcommand takes the general form:

POT n . . . . . pot/distance, pot/direction, (STATION station)

where n is the point number assigned to the POT, and the remaining data items provide the information to locate n from the previous object as a given distance and at a given, computed, or implied direction. It is the equivalent of the Course Subcommand in Traverse. The specific forms which the subcommand can take are described in the specifications for the three types of alignments.

If the station data item is given, the value of station is assigned to the POT. If it is omitted, stationing is carried forward from the previous object.

Curve Subcommand - General Information

The Curve subcommand is used to define a horizontal curve in the alignment. The Curve subcommand is a subset of the Store Curve command with certain of the data required to define the curve being automatically carried forward from the previous object as follows:

a) The curve reference point (PB) is always understood to be provided by the previous object. If the previous object was a POT, then PB is understood to be the POT. If the previous object was a curve and TL is given for distance, the PB is understood to be the PT of the previous curve; if TTL is given, then PB is the PI.

b) The back tangent direction (DB) may be carried forward from the previous object. If the previous object was a POT and DB is not given, then DB is understood to be the same direction as that used to locate the POT. If the previous object was a curve and DB is not given, then DB is understood to be the same as the forward tangent direction of the previous curve.

Because of the carry forward feature, the Curve subcommand differs from the Store Curve command principally with respect to the reference data item. PB is never given and DB is only given if the implied reference point is to be an angle point in the alignment. In an Inverse Alignment, the a/tangent data item is also not given, being automatically provided by the next object. In essence, Curve and Store Curve differ in that Curve obtains some of the data necessary to define the curve from the adjacent objects.

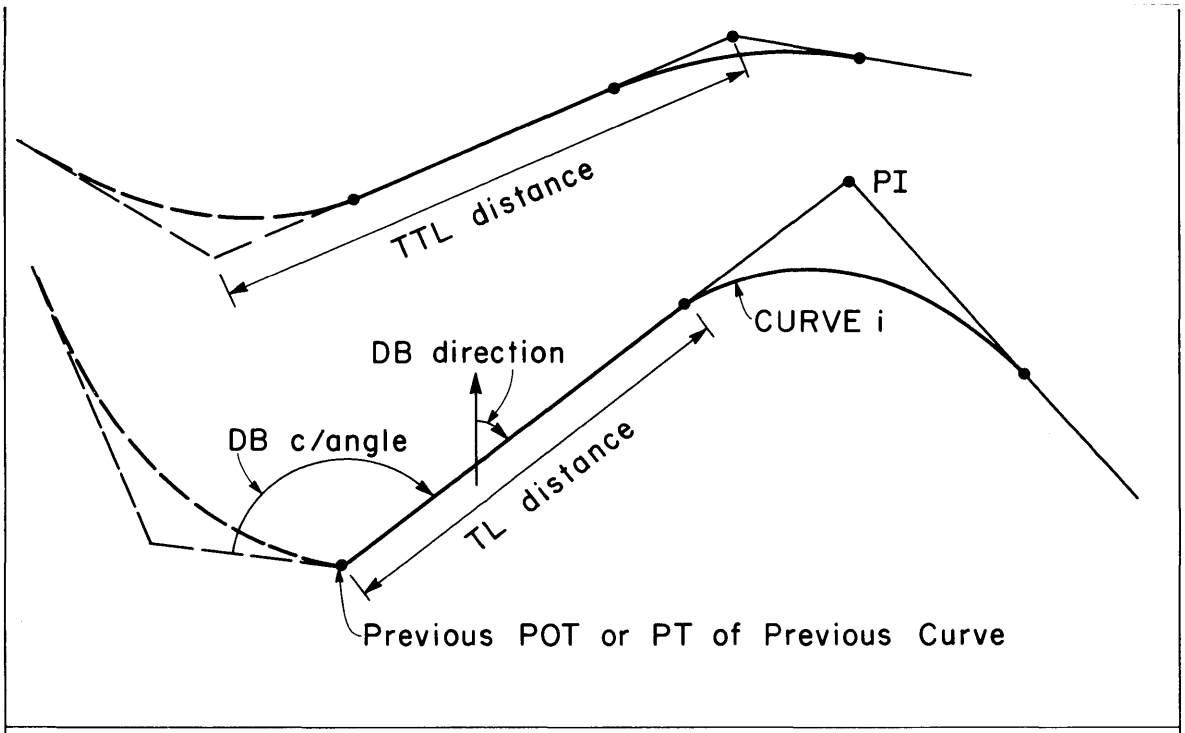
The following are additional differences between the Store Curve command and the Curve subcommand; in the subcommand

a) If c/station is not given, stationing is automatically carried forward from the previous object.

b) The print modifier is not allowed, the printing of values for all curves in the alignment being initiated by the DESCRIBE modifier.

Each curve defined by a Curve subcommand is stored in the Curve Table for later retrieval.

The specific forms for the Curve Subcommand are given in the specifications for each type of alignment.



Alignment CURVE Nomenclature and Data Items

OPEN ALIGNMENT

Introduction

In an open alignment, the data to define each curve and POT is available (no missing parts and no adjustment required). Each curve and POT is independent of those ahead of it. Subcommands are processed to completion, incrementally, in the order in which they are given. An open alignment is analogous to an open traverse.

An open alignment can always be defined by a series of Store Point, Store Curve, Locate From, and Store Chain commands. The Open Alignment command and subcommands provide a more concise way of defining and storing the alignment and the points and curves which define it.

General Form

The general form of the command and the order of the subcommands are as follows:

ALIGNMENT a, OPEN	(required)
Initial Point POT Subcommand	(required)
Curve and POT Subcommands	(optional)
Terminal Point POT Subcommand	(optional)
END of ALIGNMENT, (DESCRIBE)	(required)

Initial Point POT Subcommand

POT p/data, STATION station

The subcommand is required.

Intermediate Point POT Subcommand

POT p/data, (STATION station)
POT n TL distance (TD direction or c/angle), (STATION station)
POT n TTL distance, (STATION station)

If the second form is used and neither TD direction nor c/angle is given, a c/angle value of PLUS 180 is used. If the third form is used, the previous subcommand must have been a curve with no angle point at the PT (or ST).

Terminal Point POT Subcommand

May take any of the forms for an intermediate point subcommand and is optional. (An open alignment may end with a curve.)

Curve Subcommand

CURVE i, reference, b/spiral, element, a/spiral, a/tangent, c/station
CURVE i, STORED, c/station

where the data items are the same as for the Store Curve command except as follows:

a) the reference data item takes one of the following forms:

TL	distance
TTL	distance
DB	direction, TL distance
DB	c/angle, TL distance
PI	at pa
PI	coord (the numerical values for the PI coordinates are given)

If either of the first two forms is given, a c/angle of PLUS 180 is assumed; that is, the DB direction is carried forward from the previous subcommand (these forms should not be used for a curve immediately following the initial point POT subcommand).

b) element may be TL distance if reference gives PI and curve i does not have spirals.

c) If c/station is not given, stationing is carried forward from the previous curve or POT.

In the second form of the command, the stored curve i is used in place of a newly defined curve.

Example Open Alignment

```
ALIGNMENT 'RT-3' OPEN
POT 2 N 5000, E 6000, STA 25+00
CURVE 10, DB N 45 15 20 E, TL 1256, RADIUS 4000, P DELTA 88 27
D       12, TL 200, DEG 2 00, M 92 15
D       14, TL 300, SLB 300, R 3000, SLA 400, P 90
POT 4 TL 1000.
END OF ALIGN, DESCRIBE
```

INCOMPLETE ALIGNMENT

Introduction

An incomplete alignment is one with missing parts. The alignment must be processed as a whole to determine the values of the missing parts before the individual curves and POT's can be fully defined, computed, and stored. An incomplete alignment is analogous to an incomplete traverse with two missing parts.

An incomplete alignment can always be formulated and solved as a traverse problem. In fact, the Traverse Subprogram is used to solve for the missing parts in an incomplete alignment. The Incomplete Alignment command and subcommands provide a concise and natural way to state the problem and an automatic way of solving it.

General Form

The general form of the command and the order of the subcommands are as follows:

ALIGNMENT a, INCOMPLETE	(required)
Initial Point POT Subcommand	(required)
Curve and POT Subcommands	(required)
Terminal Point POT Subcommand	(required)
END of ALIGNMENT, (DESCRIBE)	(required)

Stationing of POT's and curves is carried forward from the given station of the initial point.

Initial Point POT Subcommand

POT p/data, TD direction, STATION station

where TD direction gives the direction of the tangent from the initial point to the POT or PI ahead.

Intermediate Point POT Subcommands

POT n, TL distance, c/angle

The data item c/angle is given if the previous POT or PT is an angle point.

Terminal Point POT Subcommand

POT p/data, TL distance, c/angle, TD direction

where TD direction gives the direction of the tangent from the previous POT or PT to the terminal point. The data item c/angle is given if the previous POT or PT is an angle point.

Curve Subcommand

CURVE i (DB c/angle), TL distance, b/spiral, element, a/spiral, a/tangent

where the data items are as follows:

spirals

SLA distance, SLB distance
SL distance

element

RADIUS distance or DEGREE angle or LENGTH distance

a/tangent

P/M DEFLECTION angle

The optional data item DB c/angle is given if the previous POT or PT is an angle point.

Missing Parts

To specify missing parts in an incomplete alignment, the following data items can take the following additional forms:

TL distance

TL ?
TL ? APPROXIMATE distance

c/angle

P/M ?
P/M ? APPROXIMATE angle

elements

R ?
R ? APPROXIMATE distance

a/tangent

P/M DEFLECTION ?
P/M DEFLECTION ? APPROXIMATE angle

TD direction

TD (AZIMUTH) ?
TD (AZIMUTH) ? APPROXIMATE direction

Each unknown distance (TL and R) counts as one missing part. The number of missing parts attributed to angle plus direction unknowns is one less than the number of unknowns. Accordingly, the combinations of unknowns which can be handled is as follows:

a) Two unknown distances and one unknown angle or direction

b) One unknown distance and two angle plus direction unknowns

c) Three angle plus direction unknowns

The APPROXIMATE form of the above data items should be used in the latter two cases (when one or both of the missing parts is an angle or direction) since two solutions are usually possible. The solution used is the one with computed values closest to the approximate value, with distance having priority over angle or direction.

Example Incomplete Alignment

```
ALIGNMENT 'RAMP-K' INCOMPLETE
POT 4 N 0. E 0., TD N 40 W, STA 0+00.
CURVE 15, TL ?  , DEG 4, SL 300, P DELTA 75
D       16, TL 0. , R 800 , P DELTA 60
D       17, TL 200, DEG 5, P DEF 55
D       18, TL 0. , DEG 3, SLA 400., P DEF ?
POT AT IP, TL ?, TD S 30 W
END OF ALIGN, DESCRIBE
```

INVERSE ALIGNMENT

Introduction

In an inverse alignment, each POT and each curve PI is at a stored point or point of given coordinates. Directions and distances between the points are computed internally by inversing. Facility is provided to adjust the coordinates of the points to make the directions between them an even value of minutes or seconds.

General Form

The general form of the command and the order of the subcommands is as follows:

ALIGNMENT a, INVERSE, round	(required)
POT Subcommand for Initial Point	(required)
Curve and POT Subcommands	(required)
POT Subcommand for Terminal Point	(required)
END of ALIGNMENT, (DESCRIBE)	(required)

The optional data item "round" takes the following general forms:

ROUND to v MINUTES
ROUND to v SECONDS

The use of the "round" data item is explained under the section on Adjustment of Inverse Alignment.

POT Subcommands

The general form for the POT Subcommand for the Initial Point, the Terminal Point, and for intermediate POT's is as follows:

POT p/data, (STATION station), (HOLD)

The first and last subcommands must be POT subcommands to define the initial and terminal points respectively. The data item "station" is required for the first (for the Initial Point). If "station" is not given for other POT's, stationing is carried forward from the previous subcommand. The optional data item HOLD is explained in the section on Adjustment of Inverse Alignment.

Curve Subcommand

The general form for the CURVE subcommand is as follows:

CURVE i, PI coord, (HOLD), b/spiral, element, a/spiral, c/station

The spirals and element data items are as described for the STORE CURVE command. The data item "c/station" is also the same except that if it is not given, stationing is carried forward from the previous curve or POT. The PI can also be given as PI at pa. (element may be TL distance if curve i does not have spirals.)

Adjustment of Inverse Alignment

If the "round" data item is given with the ALIGNMENT command, the POT's and PI's will be adjusted such that the directions of the tangents to curves and the POT courses will be to the nearest even value specified by the "round" data item. The process will be explained by an example as follows:

```
ALIGNMENT 'A' INVERSE, ROUND TO 2 MINUTES
POT 1 AT PC 25, STA 0+00
POT 2 AT 8
CURVE 5, PI AT 4, R 2000.
CURVE 8, PI N 8500  E 18600, DEG 3.5
POT 3
POT 6, N 20000  E 30000  HOLD
END OF ALIGN
```

The internal action which is taken is as follows:

a) The coordinates of the PC of curve 25 are assigned to Point 1 and are stored in the Point Table for 1, along with a station of 0+00.

b) The stored coordinates for 8 are assigned to 2. The distance and direction from 1 to 2 are computed. The direction is rounded to the nearest 2 minutes. Using the rounded direction and the computed (inversed) distance, new coordinates are computed for 2 and are stored in the Point Table for 2.

c) The stored coordinates for 4 are assigned to PI 5. The distance and direction from 2 to PI 5 are computed. The direction is rounded to the nearest 2 minutes. Using the rounded direction and computed distance, new coordinates are computed for PI 5 and are stored in the Curve Table for PI 5.

d) The given coordinates for PI 8 are assigned to PI 8. The distance and direction from PI 5 to PI 8 are computed. The direction is rounded to the nearest 2 minutes. Using the rounded direction and computed distance, new coordinates are computed for PI 8 and are stored in the Curve Table for PI 8.

e) Curve 5 is computed. (Steps c and d are actually part of the processing of Curve 5.)

f) The distance and direction from PI 8 to 3 are computed. The direction is rounded to the nearest 2 minutes. Using the rounded direction and computed distance, new coordinates are computed for 3 and are stored in the Point Table for 3.

g) Curve 8 is computed. (Step f is actually part of the processing of Curve 8.)

h) The given coordinates for Point 6 are stored in the Point Table for 6. The distance and direction from 3 to 6 are computed. The presence of the data item HOLD on the subcommand for POT 6 means that the coordinates of Point 6 are to be held and not adjusted. Accordingly, the direction is not rounded and the coordinates of 6 are not changed.

Attention is called to the fact that intermediate POT's (such as 2 and 3 in the above example) are adjusted and will become angle points if they are not already. Accordingly, intermediate POT's should not be included in an inverse alignment to be adjusted unless they are angle points.

LAYOUT COMMAND

Introduction

The Layout command is used to compute and print data for use in the field stakeout, layout, or plotting of stored objects, particularly curves and alignments.

Layout Ties

Provides layout data which ties stored points and curves to a stored point and reference direction.

LAYOUT TIES, TRANSIT at pa, (SIGHT on) pb, TIE list
LAYOUT TIES, pa TO list

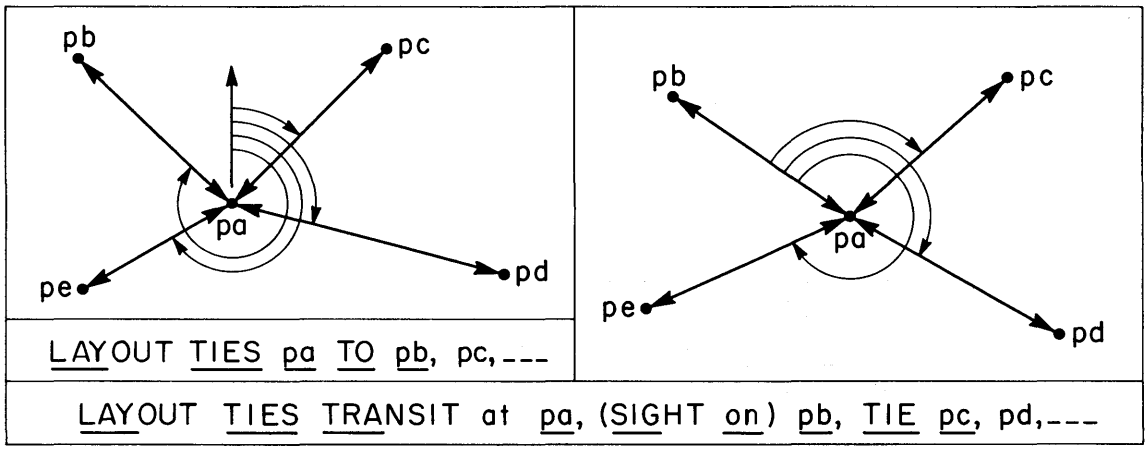
where list takes the form of a list of stored points such as

POINTS 4, 6, PC 5, 8, PT 3, 2

or is a stored chain, such as

ALIGN 'A17/B'

When a chain is given for list, the chain list is used for the list of points (each point in the chain list, the end points of each course, and the curve point for each curve is tied).



The layout data which is printed for each point in the list includes the following:

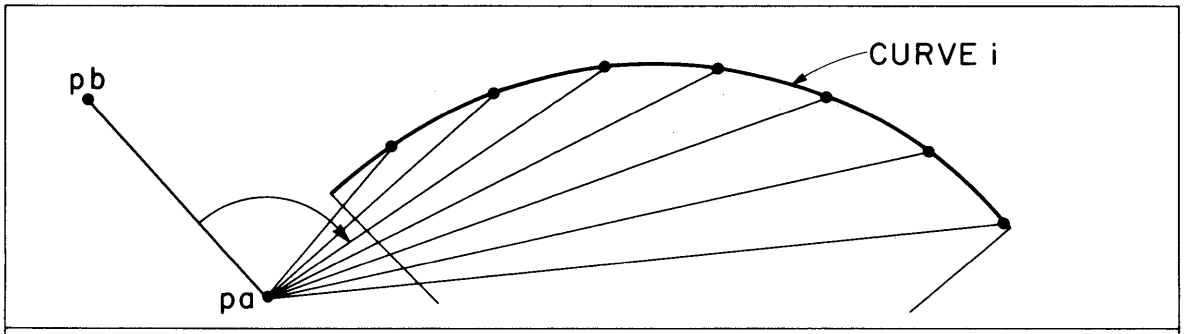
a) distance from pa to the point

b) clockwise angle at pa from pb to the point

c) counterclockwise angle at pa from pb to the point

d) direction (azimuth or bearing) from pa to the point

When the second form of the command is given, only the distance and direction are printed.



LAYOUT CURVE i (TRANSIT at) pa, (SIGHT on) pb, (EVEN) v (STATIONS)

Layout Curves

Provides layout data for stored curves.

LAYOUT CURVE i, transit, even, off, plus
LAYOUT CURVES, ALIGNMENT a, even, off
LAYOUT ALL CURVES, even, off

where the data items take the following forms:

data item	form	
transit	(TRANSIT at) pa, (SIGHT on) pb	
even	(EVEN) v (STATIONS)	
off	(OFFSETS)	
plus	PLUS STATION va, (STATION vb), ...	the two forms
	PLUS pa, pb, pc	may be mixed

If transit is not given, pa is taken to be the PC of curve i and pb is taken to be the PI of the curve (this will always be the case in the second two forms).

Layout data is computed for each even station, each plus station and point, and each curve point, the following being printed:

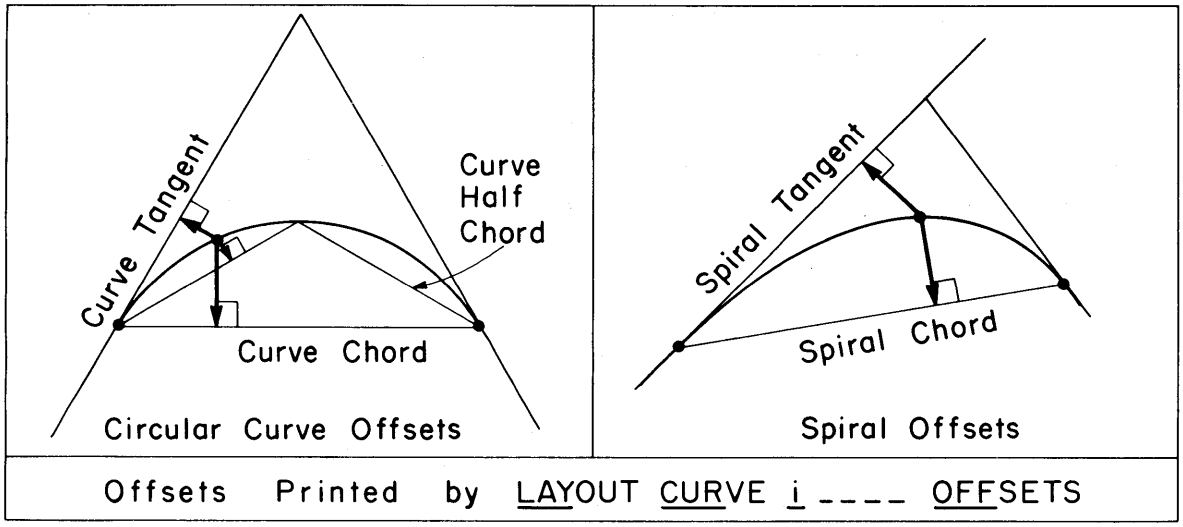
a) station of the point
b) clockwise angle at pa from pb to the point
c) counterclockwise angle at pa from pb to the point
d) deflection angle at pa from pb to the point
e) chord length from the previous point
f) distance from pa to the point

If the optional word OFFSETS is given, three offset distances are also computed and printed for each even station as follows:

a) offset from the closest tangent
b) offset from the long chord
c) offset from the half chord (chord to circular curve midpoint)

If the curve has spirals, the back spiral, the circular curve portion, and the ahead spiral are treated separately and layout data for each is produced. If transit is not given, for the back spiral pa is taken to be the TS and pb the PI of the total curve; for the circular curve, pa is taken to be the SC and pb the PI of the circular curve; for the ahead spiral, pa is taken to be the ST (backed in) and pb the PI of the total curve. Offsets are computed for the spiral or circular curves on tangents and chords, not those of the total curve.

In the second and third forms of the command, the layout data is printed for all curves in Alignment a and for all stored curves, respectively. The layout report is preceded by a printout of the curve elements and curve points for the curve (the equivalent of a PRINT CURVE i command). Accordingly, the output is a rather complete report on the curve for sending to the field for stakeout activity.



Offsets Printed by LAYOUT CURVE i ____ OFFSETS

Layout Offsets

Provides layout offsets from one stored chain (baseline) to another stored chain (alignment).

LAYOUT OFFSETS, (BASELINE) a to (ALIGNMENT) b, list

where list is one or more of either of the following two forms or a mix of the two:

(STATIONS) va, vb, vc, ...
EVEN v (STATIONS) va TO vb

The following is an example:

LAYOUT OFFSETS, BASELINE 'A4' TO ALIGNMENT 'R3', 6+25, 14+50, EVEN 50 STA 20+00 TO 30+00, 35+27.16, EVEN 100 40+00 TO 50+00, 68+23, 100+00

a and b are the names of any stored chains (any of the chain labels may be used in place of BASELINE and ALIGNMENT). The offset lines are perpendicular or radial to a and are computed for given stations and/or even stations on a. The computed and printed values for each offset include:

a) station on a

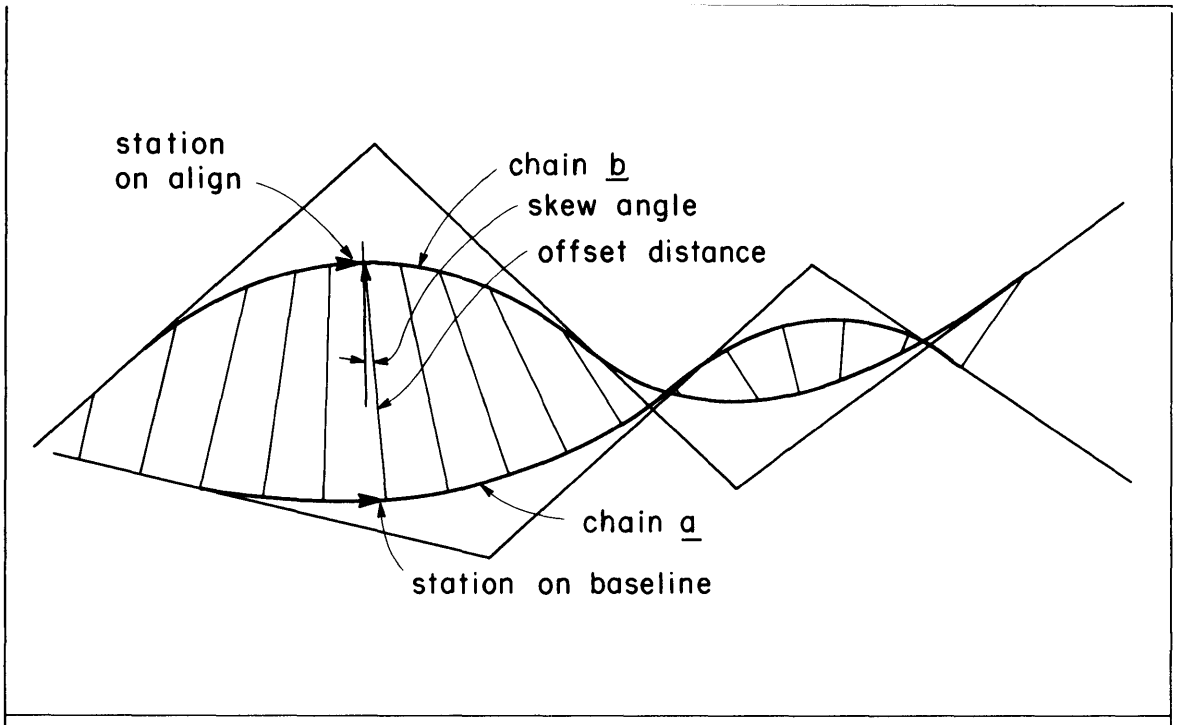
b) offset distance from a to the intersection of the offset line with b (minus if b is to the left of a)

c) station on b of the intersection

d) skew angle at the intersection

Offsets will not be computed beyond the limits of a or b. If a given station is not on a, a message is printed. If the station occurs more than once (due to station equations), the offset will be computed for the first occurrence of the station ahead of the previous given station. In the Even Station form, if va is less than the point at the beginning of a, the beginning point is used for va. The increment is added repeatedly until either vb or the end point of a or b is reached. If a has station equations such that va and/or vb can occur more than once, the first occurrence of va and the first occurrence of vb ahead of va are used.

If the station values in list are given in order of increasing magnitude, the speed of processing is greatly increased. There is one restriction on the command. The processing routines assume that each offset intersection is ahead of the previous one. This will always be the case as long as the center of no curve on a falls between a and b.



LAYOUT OFFSETS, CHAIN a to CHAIN b, EVEN v (STATIONS), v, v,...

Layout Coordinates

The command is used to compute and print the coordinates at even stations on a stored alignment.

LAYOUT ALIGNMENT a, COORDINATES at (EVEN) v (STATIONS)

In addition to the coordinates of each even station, the station and coordinates of each point and curve point for each curve in the chain list for a are printed.

Example Layout Commands

```
LAYOUT TIES, TRANSIT AT 4, SIGHT ON 8, TIE 3, 5, 7, 9
LAYOUT TIES, POINT 5 TO 8, 10, 12, 14, 13, 11
LAYOUT TIES, 28 TO ALIGN 'RAMP-J'
LAYOUT CURVE 18, TRANSIT AT 23, 25, 100 STA
LAYOUT CURVE 9, EVEN 10 STA, OFFSETS, PLUS STA 15+76.15
LAYOUT CURVES, ALIGNMENT 'RT-3', EVEN 50, OFFSETS
LAYOUT OFFSETS, 'RT-3' TO 'RAMP-J', EVEN 100
LAYOUT ALIGN 'CL/8', COORD AT EVEN 20 STATIONS
```

TRAVERSE COMMAND

Introduction

The Traverse command and associated subcommands can be used to solve a wide variety of traverse problems, ranging from the very simple to the very complex. The traverse may be open or closed. If closed, it may be complete or incomplete. If incomplete, it may have any combination of one or two missing parts. If closed with one or no missing parts, the traverse will be adjusted, with a choice of adjustment rules. It may be an angle traverse, direction traverse, or combination (mixed). Individual courses may be held fixed during adjustment. The traverse is automatically stored in the Chain Table for later retrieval, and each traverse point is stored in the Point Table.

General Form

The general form of the command and the normal order of the subcommands are as follows:

TRAVERSE a	(required)
Adjust Subcommand	(optional)
Closure Subcommand	(optional)
Back Subcommand	(optional)
Course Subcommands	(2 or more required)
Ahead Subcommand	(optional)
END of TRAVERSE, (REPORT), (PRINT), (SKETCH)	(required)

where a is the name of the traverse for storage in the Chain Table. The block of Course subcommands must be in the order of the courses, but the other subcommands can be in any order as long as they come after TRAVERSE and before END. The optional subcommands are not required for many types of problems, and in others, standard values are used if they are not given.

If REPORT is given, a complete corrections report is printed for adjusted traverses which gives unadjusted, correction, and adjusted values for all distances, directions, angles, latitudes, departures, and point coordinates. If PRINT is given, tables of adjusted course values (distances, angles, and directions) and point coordinates are printed. If SKETCH is given, a sketch of the traverse is plotted on the printer. The sketch is useful to spot gross mistakes in the traverse.

Adjust Subcommand

ADJUST by rule

where rule is one of the following:

LINEAR rule
TRANSIT rule
COMPASS rule
CRANDALL rule
LEAST SQUARES, weight

and where "weight" is one of the following:

TAPE (WEIGHT) v
DISTANCE (WEIGHT) v

and v is the numerical value of the weight to be assigned to each 100-foot tape length (if TAPE is given) or each course length (if DISTANCE is given) relative to unit angle weight. If "weight" is not given, DISTANCE WEIGHT 1. is used (distances and angles are given equal weight).

If the traverse requires adjustment but the Adjust subcommand is not given, the standard rule which is used is the equivalent of giving the following:

ADJUST BY LEAST SQUARES, DISTANCE WEIGHT 1.

Closure Subcommand

CLOSURE plan, angular

where "plan" takes the form: ONE (PART IN) v
where "angular" takes the form: PER (ANGLE) v (SECONDS)

The closure values are used as a specification which the actual closure of a closed traverse is checked against. If the traverse does not close within twice the specification, processing of the traverse continues but no entries are made in the Data Tables. The "angular" specification is used in the preliminary angle adjustment, and the "plan" specification is used in the general adjustment.

If the traverse requires adjustment but the Closure subcommand is not given, the standard closure specification which is used is the equivalent of giving the following:

CLOSURE ONE PART IN 2000, PER ANGLE 60 SECONDS

The meaning of the command is that the preliminary angle adjustment per angle must not exceed 60 seconds and the traverse must close within one part in 2000 for the traverse to be within specifications. If it does not close within twice these values (one part in 1000 and 120 seconds per angle), it is assumed that there is a bust in the traverse and no entries should be made in the Data Tables.

A complete report on the closure of the traverse is printed.

```
**********************************************************************
CLOSED TRAVERSE
**********************************************************************
PRELIMINARY ANGLE ADJUSTMENT
   ANGULAR ERROR OF CLOSURE REPORT
         ERROR PER ANGLE    1 57 30.00
         TOTAL ERROR        7 50  0.00
WARNING-ANGLE CLOSURE EXCEEDS SPECIFICATION
CLOSURE EXCEEDS TWICE SPECIFICATION
WARNING-NO ENTRIES WILL BE MADE IN DATA TABLES FOR      THIS TRAVERSE
**********************************************************************
ONE MISSING DISTANCE IN TRAVERSE
UNADJUSTED VALUE FOLLOWS
   DISTANCE FROM POINT     6    TO POINT     8  3735.825
TRAVERSE WILL BE ADJUSTED
**********************************************************************
            CLOSURE REPORT - PLANIMETRIC

       ERROR IN X 54.807  TOTAL ERROR     215.013 PERIMETER   11855.672
       ERROR IN Y207.911  CLOSURE RATIO     55.14 FIXED PERIM      0.0
       DIRECTION OF CLOSING LINK 194 46   3.45   TOTAL PERIM 11855.672
WARNING - CLOSURE EXCEEDS SPECIFICATION
CLOSURE EXCEEDS TWICE SPECIFICATION
PROCESSING CONTINUES BUT NO ENTRIES WILL BE MADE IN     DATA TABLES
**********************************************************************
   COMPASS RULE ADJUSTMENT

            CORRECTION TABLES
                          UNADJUSTED   CORRECTION ADJUSTED VALUE

            DISTANCE TABLE

  DISTANCE     4 TO     6     4521.727      78.742       4600.469
  DISTANCE     6 TO     8     3735.825      -1.337       3734.489
  DISTANCE     8 TO     2     3598.120     -48.970       3549.149

            AZIMUTH TABLE

  AZIMUTH  BACK          341 35  0.00
  AZIMUTH     4 TO     6    7 30.00   -          30 50
              6 TO          0.00
```

Back Subcommand

BACK direction

The subcommand is used to give a backsight direction at the initial point of the traverse for use in computing course directions from input angles and/or to provide the basis for making a preliminary angle adjustment. The subcommand may be omitted if it is not needed for forward computation of course directions or if no preliminary angle adjustment is to be made. If the subcommand is not given, a Back direction of due North is assumed.

Course Subcommand

COURSE a (j TO) n (FIXED) c/distance, c/direction

If optional data item a, the name of the Course, is given, the Course will be stored in the Course Table. If optional word FIXED is given, the length and direction of the course will be held fixed and will not be changed during general adjustment.

The data item j is the beginning point of the course and n is the end point. Point j must be the same as Point n of the previous Course subcommand. j may be omitted except for the first Course subcommand where it is the initial point of the traverse. n of the last Course subcommand is the terminal point of the traverse.

The c/distance data item takes any one of the following forms:

distance	(standard data item)
(DISTANCE) ? (APPROXIMATE distance)	
(DISTANCE) SAME	

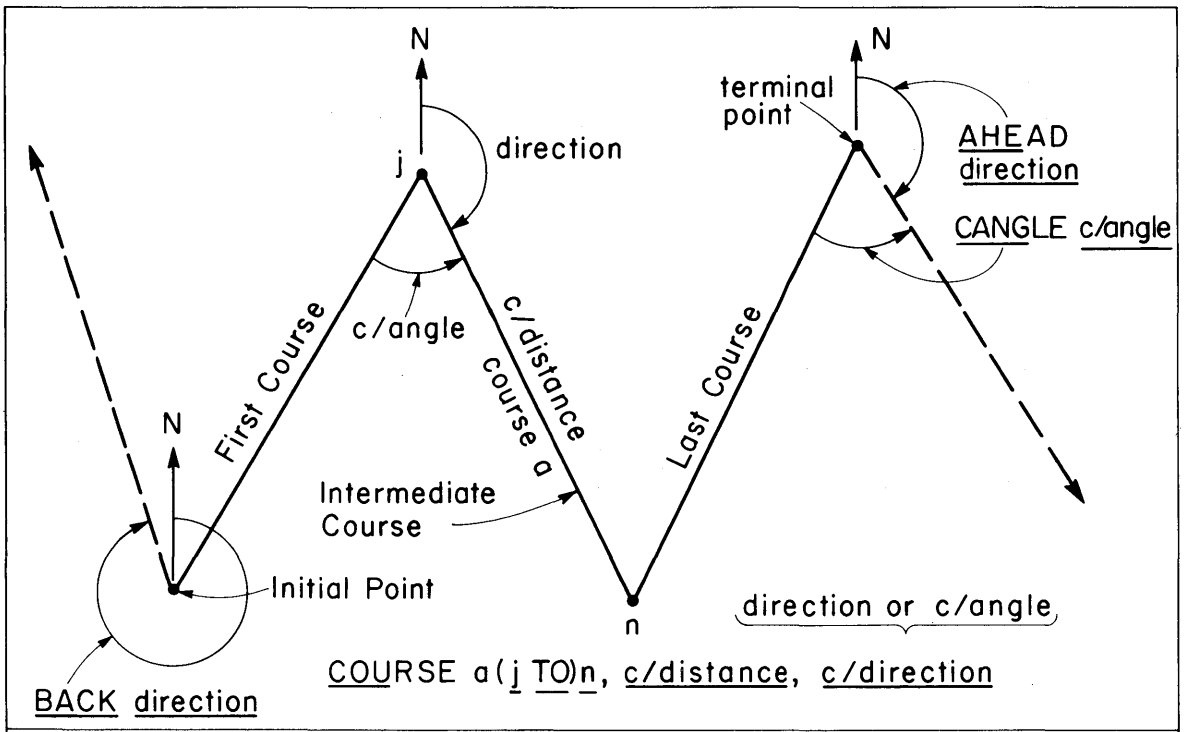
The second form is used when the length of the course is unknown and is to be treated as a missing part. The optional APPROXIMATE data item is used to provide an approximate value for the unknown distance in two solution cases. The SAME form is used when the length of the course is to have the same value as that of the previous unknown distance, the two to count as a single unknown in a missing parts traverse.

The c/direction data item takes any one of the following forms:

direction	(standard data item)
(AZIMUTH) ? (APPROXIMATE direction)	
c/angle	(standard data item)
ANGLE ? (APPROXIMATE P/M angle)	

The first form is used when the direction of the course is given as input data, the second when the direction of the course is unknown. The third form is used when the angle at j from the previous course to n is given as input data, the angle to be used internally to compute the course direction. The fourth form is used when the c/angle is unknown. The optional APPROXIMATE data item in the second and fourth forms is used to provide an approximate value for the unknown for use in two solution cases.

The word UNKNOWN may be used in place of the symbol ? in the c/distance and c/direction data items and should be used when the problem is to be run at an installation which does not have the ? character on its printer chain.



TRAVERSE DATA ITEMS AND NOMENCLATURE

Ahead Subcommand

AHEAD direction, CANGLE c/angle

The subcommand is used to give a foresight direction and closing angle at the terminal point of the traverse for use in computing course directions from input angles and/or to provide the basis for a preliminary angle adjustment. The Ahead direction is held fixed in making the adjustment. The c/angle data item is to give the closing angle at the terminal point from the backsight (next to last point) to the foresight direction. It may take any of the angle forms for the c/direction data item including the angle unknown form. The subcommand may be omitted if it is not needed for back computation of course directions or if no preliminary angle adjustment is to be made. If the subcommand is not given, an Ahead direction of due North is assumed, and the closing angle is treated as an unknown.

Closed Traverse

A traverse is processed as a closed traverse if both the initial point and terminal point are defined points (stored in the Point Table). They may be the same point (traverse closed on itself). If they are the same point but the point is not defined, it is assigned 0., 0. coordinates and the traverse is processed as a closed traverse. A closed traverse may be an angle or a direction traverse, and it may have no, one, or two missing parts. If it has no or one, it is always adjusted.

Open Traverse

A traverse is processed as an open traverse if it is not closed. Either the initial point or the terminal point should be a stored point. If neither point is stored, the initial point is assigned 0., 0. coordinates. An open traverse may be an angle or a direction traverse, but it cannot have missing parts and no general adjustment is made (a preliminary angle adjustment is possible).

Direction Processing

Due to the wide variety of ways in which course directions can be given, the traverse is converted to a basic form for internal processing of angles and directions and for computation of unknowns. The basic form works with the initial angle (the angle at the initial point from due North to the first course), the angle at each intermediate course point, and the terminal angle (the angle at the terminal point from the last course to due North).

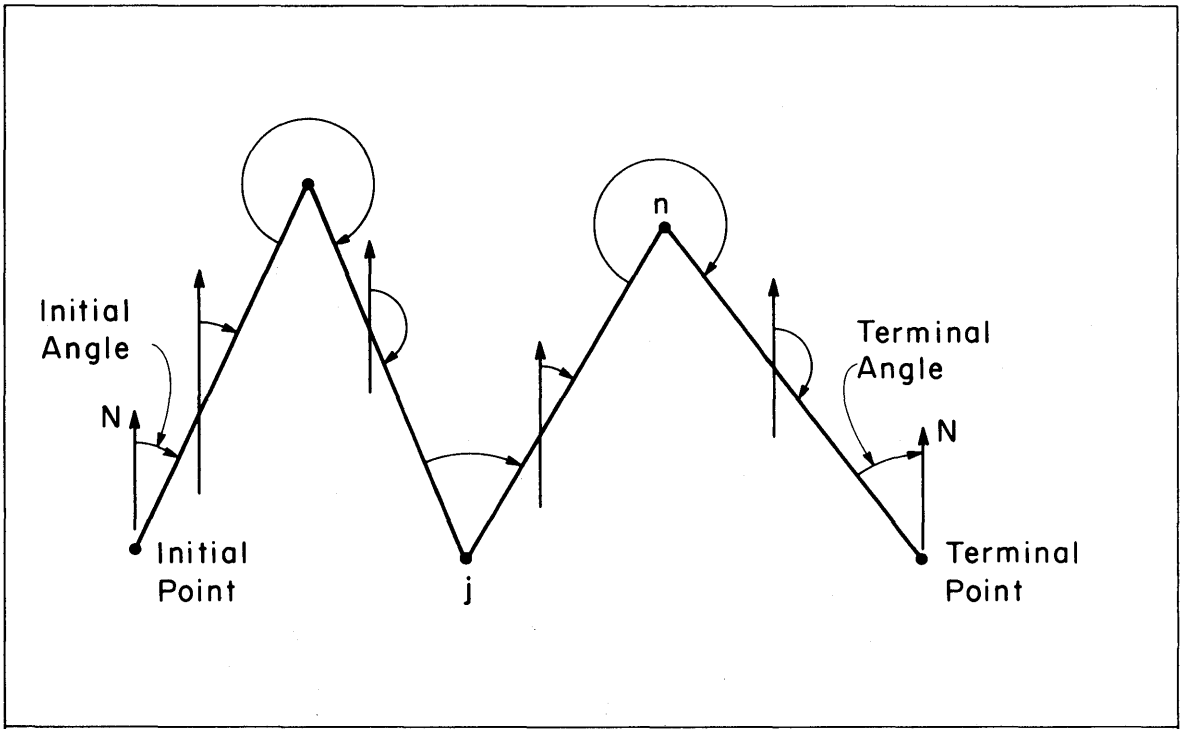
The basic form angles are determined by the following stages of processing:

1) Starting with the given (or assumed) Back direction, directions are carried forward, and all angles which can be computed by a difference in directions are determined.

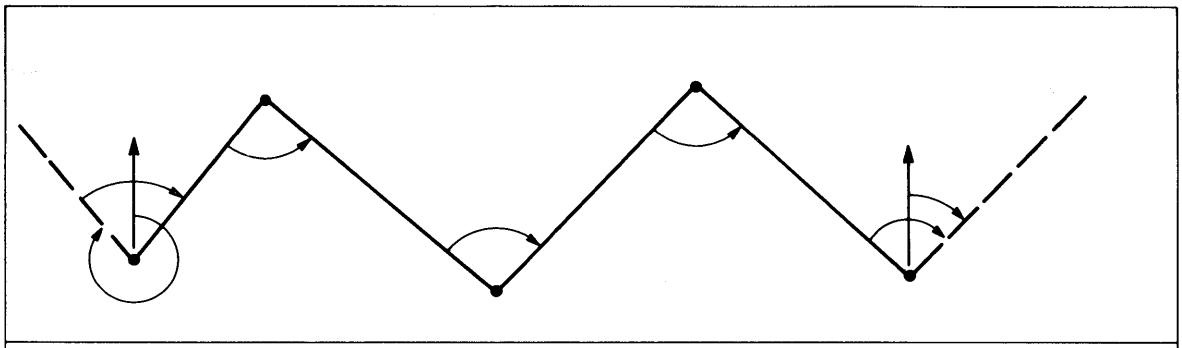
2) If all angles cannot be computed from forward computation, then directions are carried backward from the Ahead direction, and all angles which can be computed by a difference in directions are determined.

3) If all angles cannot be computed by a combination of forward and back computation, then one or more angles constitute missing parts in an incomplete closed traverse. The unknown angles and any unknown distance are computed.

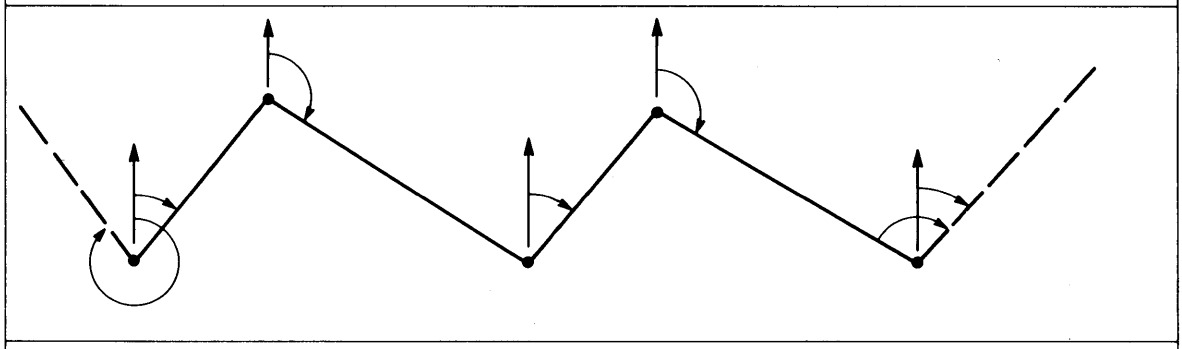
After numerical values are determined for all of the basic form angles and the preliminary angle adjustment made (if required), the unadjusted course directions are computed, and traverse processing continues.



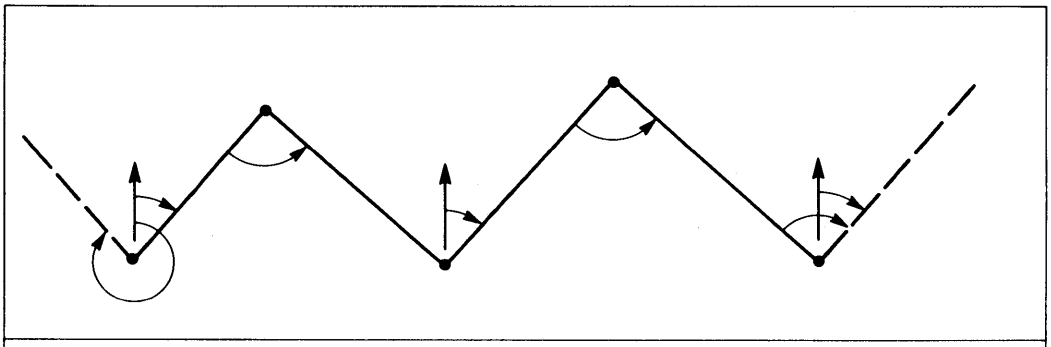
BASIC FORM OF TRAVERSE FOR INTERNAL PROCESSING



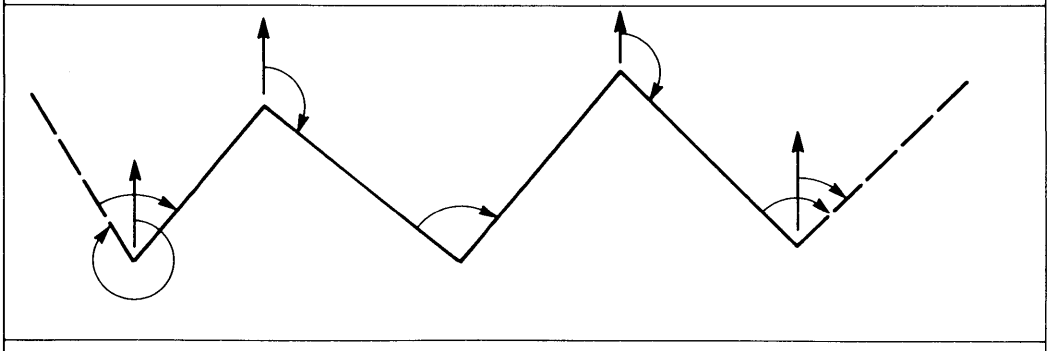
ANGLE TRAVERSE



DIRECTION TRAVERSE



MIXED TRAVERSE A



MIXED TRAVERSE B

Preliminary Angle Adjustment

A preliminary angle adjustment will be made when the Ahead subcommand is given, the closing angle is given, and the closing angle can also be computed by forward computation. The difference between the computed value and the given value is treated as an angular error of closure. The angular error of closure is checked against the angular closure specification, and a preliminary angle adjustment report is printed. If the closure is within twice the specification, the angular error of closure is linearly distributed, each basic form angle receiving an equal correction. If there are any FIXED courses in the traverse, their direction will be affected by the preliminary angle adjustment but not by the general adjustment.

Unknowns

All geometrically solvable cases of unknowns in a closed traverse are automatically handled. The Engineer does not have to identify the particular case represented by the traverse as the internal logic determines the case. Any solvable combination of one or two geometric missing parts can be handled. Since an unknown angle and, in some cases, an unknown direction can be determined by forward and back computation, the number of geometric missing parts can be less than the number of unknowns (? form).

In terms of the basic form angles, the following cases of unknowns can be handled internally:

a) One unknown angle
b) Two unknown angles
c) Three unknown angles
d) One unknown distance
e) Two unknown distances
f) One unknown angle and one unknown distance
g) One unknown angle and two unknown distances
h) Two unknown angles and one unknown distance

The cases are identified internally during direction processing as previously described. One unknown angle is usually eliminated as an unknown during forward and back computation.

To assist in identifying the allowable cases externally, two typical types of traverses, angle traverse and direction traverse, are defined. An angle traverse is one in which Back and Ahead are given, and all c/direction data items are given in the c/angle or unknown angle form. The allowable cases

are the same as those described above for the basic form. A direction traverse is one in which the c/direction data items are given in the direction or direction unknown form. The allowable cases are

a) One unknown direction
b) Two unknown directions
c) One unknown distance
d) Two unknown distances
e) One unknown direction and one unknown distance

If the Ahead subcommand is given, and if the closing angle is given as unknown, that unknown angle can be in addition to the above unknowns.

Most traverses will be of a mixed type; that is, some of the c/direction data items will be of an angle form, others of a direction form, and various combinations of Ahead and Back being given or not given. The system expects the mixed type and converts all, including angle and direction, to the basic form for internal processing.

Two Solution Cases

The computation of unknowns when there are two missing geometric parts is handled internally by formulating and solving the equivalent intersection problem. Two unknown distances are the equivalent of intersecting two lines. If the lines are not parallel (no solution), a single intersection (solution results). When one or both of the missing parts are an angle or direction, two solutions can result as follows:

(1) Unknown Distance and Direction or Angles

The following cases are the equivalent of intersecting a line and a circle:

1 unknown distance and 2 unknown angles
1 unknown distance and 1 unknown direction

The line will intersect the circle at two points (if it intersects at all), yielding two solutions for the unknowns.

An approximate value for the unknown distance should be given in the distance data item. The solution closest to the approximate value will be used. If an approximate distance is not given, an approximate value for the unknown direction or one of the unknown angles should be given. If both approximate distance and approximate direction or angle are given, approximate distance is used, the other approximate values being ignored. If no approximate value is given, the distance solution with the smallest positive value is used.

(2) Unknown Angles and/or Directions

The following cases are the equivalent of intersecting two circles:

3 unknown angles
2 unknown directions
1 unknown angle and 1 unknown direction

The two circles will intersect at two points (if they intersect at all), yielding two solutions for the unknowns.

An approximate value for one of the unknown angles or directions should be given in the c/direction data item. The solution closest to the approximate value will be used. If more than one approximate value is given, the first one is the one used, the others being ignored. If no approximate value is given, the solution with the circle intersection nearest to the origin of coordinates is arbitrarily used.

If approximate values are given in cases which do not have two solutions, the approximate values are ignored.

Unsolvable and Partially Solvable Cases of Unknowns

Certain geometric conditions in a closed traverse can result in situations where the unknowns cannot be computed or can only be partially computed. The system checks for such conditions and notifies the Engineer via appropriate messages. The following are examples of such conditions:

1. Unknown Distance Cases

   a. Two unknown distances are parallel, having equal or opposite directions.

   b. SAME unknown distances combine with unknown distances such that composite directions are parallel or cancel. (Partially solvable in some cases)

2. Unknown Angles - Basic Form

   a. Two unknown angles are for first course and closing angle when traverse closes on itself. (Solved assuming first angle is 180 degrees)

   b. Three unknown angles result in a no intersection case due to mistakes in input data.

Generalized SAME

In the Course subcommand, the (DISTANCE) SAME form of the c/distance data item can be used to define adjacent or separated unknown distances which are to be solved as equal. Such a "matched pair" is equivalent to one missing part. Thus two such matched pairs can be solved, or one pair and another unknown distance, when two unknown distances are allowed. This feature is provided so that the radius (or tangent) of a circular curve, appearing as an unknown in two courses, can be treated as a single unknown.

The rule of order is that the first SAME is associated with the first unknown distance, the second SAME with the second unknown distance. If the matched pair is separated by another unknown distance or another matched pair, the rule of order can be reversed by the following expedient: if one or both of the unknown distances are given in the form

(DISTANCE) ? APPROX -1.

that is, the approximate distance value is a negative arbitrary value, the rule of order is then reversed. The first SAME is then associated with the second unknown distance, and the second SAME (if present) will be associated with the first unknown distance. Any configuration of matched pairs and unknown distances may be specified by using or reversing the rule of order.

General Adjustment

A general adjustment is made in the case of a closed traverse with no unknowns or in the following cases of unknowns (single missing part):

One angle unknown
One direction unknown
One distance unknown
Two angles unknown
One angle unknown and one distance unknown

The traverse is adjusted by the rule given in the Adjust subcommand.

```
COGO
$          EXAMPLE PROBLEM T1
$     OPEN TRAVERSE/BEGINNING POINT GIVEN
STORE 3  N 25763.17  E  43792.98
TRAVERSE 'T1'
COURSE 3 TO 8   4472.15   N 26 30 E
D           4   3162.35   S 18 20 E
D           6   3605.84   N 33 20 E
D           2   4123.15   S 14  5 E
D           5   3163.42   N 18 25 E
END OF TRAV,PRINT

COGO
$          EXAMPLE PROBLEM T2
$     OPEN TRAVERSE/TERMINAL POINT GIVEN
$      PRELIMINARY ANGLE ADJUSTMENT
STORE 12   5000   8000
TRAVERSE 'T2'
BACK S 44 23 W
CLOSURE PER ANGLE 120 SECONDS
COURSE 'A' 25 TO 18  3125   M 116 30
D    'B'     15 3610  P 142 15
D    'C' 15 TO 9 3580  M 23 28
D    'D'     12 2190  M 84 17
AHEAD N 45 57 W, CANGLE M 72 33
END TRAVERSE,PRINT

COGO
$          EXAMPLE PROBLEM T3
$     CLOSED TRAVERSE
STORE 24  0 0
TRAVERSE 'T3'
COURSE 24 TO 28  2816.87  N 43 28 E
D           17  4112.73  M 122
D           35  2237.15  M 131 29
D           99  FIXED 4123.76  S 76 08 W
D     24  3718.52  M 128 30 00
END TRAV,REPORT,SKETCH

COGO
$          EXAMPLE PROBLEM T4
$      CLOSED TRAVERSE,ONE MISSING PART
$        PRELIM. ANGLE ADJUST. REQUIRED
STORE  4 N 1000 E 2000
D      2 N 2000 E 5000
TRAVERSE 'T4'
ADJUST BY COMPASS RULE
CLOSURE ONE IN 3000
BACK  N 18 25 W
COURSE 4 TO 6 4521.73  P 51 30
D           8  DIST ?   P 257 15
D           2  3598.12  M 48 40
AHEAD S 72 10 E, CANG   M 306
END TRAV,REPORT
```

```
COGO
$          EXAMPLE PROBLEM T5
$     TWO MISSING DISTANCES, ONE UNKNOWN ANGLE
STORE  46    N 2500    E 1000
D      18    N 1000    E 3500
TRAVERSE  'T5'
BACK 46 TO 18
COURSE  46 TO 32 DIST UNKNOWN    ANGLE MINUS 77 00
D              25        2500.            M      98
D           78     3150.     ANG  UNKNOWN
D              52        3620             M      56
D              18      UNKNOWN            M      97
AHEAD 18 TO 46, CANGLE MINUS 157 0 0
END OF TRAVERSE,PRINT,SKETCH
COGO
$          EXAMPLE PROBLEM T6
$     THREE UNKNOWN ANGLES
STORE  4     N 1500    E 3000
D      5     N 1500    E 9000
TRAVERSE 'T6'
BACK  N 26 34 W
COURSE  4 TO 3  3612.57 ANGLE UNKNOWN APPROX P 60
D      3 TO 8  4121.37  ANG  UNKNOWN  APPROX P 300
D       8 TO 2  3574.16     P  47 28 0
D        2 TO 5  2212.95 ANG  UNKNOWN
END TRAV
COGO
$          EXAMPLE PROBLEM T7
$     COMPUTED AND FIXED COURSE
STORE  24  N 4000 E 5000
D      18  N 5000 E 2000
LOCATE 33 FROM 24   2215. P 135 FROM 18
TRAVERSE 'T7'
COURSE 33 TO 27  4150.  S 14 E
D              38  4130.  N 76 W
D              21  3150.  S 71 W
D              35  3135.  N 18 E
D      35 TO 15  FIXED DIST 18 TO 24, AZ 18 TO 24
D              33  DIST UNKNOWN  N 45 E
END OF TRAVERSE PRINT
PRINT COURSE 18 TO 24, 35 TO 15
COGO
$          EXAMPLE PROBLEM T8
$     ILLUSTRATION OF USE OF SAME UNKNOWN DISTANCE CAPABILITY
STORE POINT 5    N 5000    E 1000
D             2    N 4000    E 1700
D            12    N 2300    E 8000
TRAVERSE 'T8'
COURSE  2 TO 4  982.           5 TO 12
D              7  DIST UNKNOWN  PLUS   90
D              3  SAME          MINUS  112 0 0
D              6  720.          PLUS   90
)              8  DIST UNKNOWN  MINUS  90
)             10  SAME          PLUS  115 0 0
)             12  1315.         MINUS  90
END TRAV PRINT
```