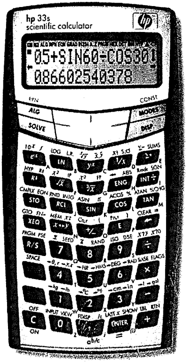
B0001	LBL B	[↰] [+] [B]	Begin label
B0002	CLVARS	[↰] [←] [2]	Clear variabl…
B0003	INPUT B	[↰] [0] [B]	Prompt for bear…
B0004	INPUT Q	[↰] [0] [Q]	Prompt for Qua…
B0005	x<>y	[x↔y]	Swap
B0006	→HR	[↰] [5]	Convert to D.dd
B0007	x<>y	[x↔y]	Swap
B0008	ENTER	[ENTER]	Enter
B0009	ENTER	[ENTER]	Enter
B0010	2	[2]	2
B0011	÷	[÷]	Divide
B0012	IP	[↱] [x^2]	Get integ…
B0013	π	[↱] [COS]	Pi
B0014	→DEG	[↰] [6]	Radians t…
B0015	x	[×]	Multiply
B0016	x<>y	[x↔y]	Swap
B0017	LASTx	[↰] [LASTx]	Return la…
B0018	x	[×]	Multiply
B0019	COS	[COS]	Cosine
B0020	R↑	[↱] [R↓]	Roll regi…
B0021	x	[×]	Multiply
B0022	-	[−]	Subtract
B0023	→HMS	[↱] [5]	Convert t…
B0024	RTN	[↱] [+]	End

CK=506F
LN=84

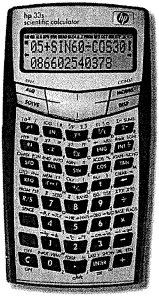


Surveying Solutions for the hp33s

2nd Edition

D'Zign

Surveying Solutions for the hp33s



2nd Edition

Manufactured in the United States of America

First printing December 2005

published by **Software by** D'Zign
Tollhouse, California, U.S.A.

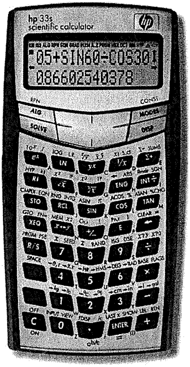
10 9 8 7 6

acknowledgements
Special thanks to Luan Le, beta-tester and proof-reader, for his excellent help in preparing this book . We would also like to thank the LSIT Review Class members from the City of San Jose, California Surveying Section for volunteering as guinea pigs, and for their input, and Barry Ng, Former San Jose City Surveyor, for arranging the classes.

contents

description	text page	Program steps
program cautions and comments	**1**	
direction of a line	2	
Azimuth to Bearing	4	3
Bearing to Azimuth	5	4
add-subtract Degrees, minutes and seconds	6	5
circular curves	8	P1
circular curve layout	9	P3
triangle solutions	14	P4
vertical curves and grades	21	P8
station with elevation known	24	P9
vertical intersections	24	P9
coordinate geometry	26	
traverse and closure	28	P11
inversing used separately	30	P12
used during a traverse	31	
for stakeout calculations	32	
intersections	33	P13
calculating missing sides	37	
line/curve to curve intersections	40	
slope staking as intersection	40	
equations and the NCEES test, solver	41	
trouble-shooting your programs	41	
answer key	42	
LBL U for undoing flag settings	45	
program LN and checksum chart	**46**	

This book has been developed for two reasons. The first is that those doing engineering or surveying calculations need access to an inexpensive but functionally programmable calculator to do everyday calculations. The second is that NCEES tests no longer allow the use of any calculators that have alpha or memory storage capabilities and the hp 33s is the best of the allowed calculators that we found.



While it is programmable, it has limits. It has a lot of memory, but can only hold a total of 26 different program variables (A through Z), and only has a few flag settings available for branching. It is also limited to only the same 26 storage registers for most programming, which prohibits storing coordinates.

This book is written as a review course workbook, as well as a solution book for programming. If all you are after is the programming, fine. For those getting ready to take one of the exams, we have included review information and some practice problems. Most of the problems are similar to the same type of problem found in the NCEES tests. In the rest of this workbook we will review some of the basics of whatever the subject is we're working on, maybe do a few exercises, then look at shortcut methods with the calculator. After that, we can look at how a program works to do the same solution.

Some programming cautions and comments:

When you press [⬑] [←] to clear the stack or registers, NEVER use option 3ALL By stroking [3]. That would clear everything you've already done.

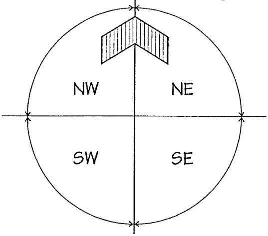
In our programs we will use an equation to actually spell out what is needed as the prompt instead of the calculator's INPUT function; you will have enough to keep track of during a test without having to remember what the single-letter prompt is asking for (step A0003 for example, could have just prompted A?), The actual keystrokes for input of the step are:

[⬏] [STO](EQN) [RCL] [A] [RCL] [Z] [RCL] [I] [RCL] [M] [RCL] [U] [RCL] [T] [RCL] [H] [R↓](COLON) [ENTER]

Our notation in the program steps will be [⬏] [STO] *then* [RCL] *before each alpha input* and means that you will stroke [RCL] before each *alpha* key.

Another thing about some of these programs, which flow from one label to another. LBL A flowing directly to LBL N without any STOP or RTN to end LBL A is one example. If a program is listed in this manner in the program pages it means that the whole group is input together. Other examples of this occur in LBL 0, curve layout, which flows directly into LBL Q, LBL V flows into LBL Y, LBL Z into LBL X, and LBL T flows into LBL J. These must be input continuously as a single program as we will do in LBLs A&N on page 4.

Directions of a line are a good place to start. In surveying within the continental United States we use two primary types of directional notation, *azimuths* and *bearings*, to tell us what direction a line is going. All record maps and legal descriptions are usually done using ***bearings*** (also sometimes called quadranted bearings) with the compass divided into four 90° quadrants of direction as shown to the left.

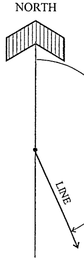


Directions by Bearings

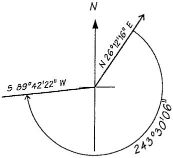
In the northerly quadrants the bearing angle is measured from the North; going *Clockwise for the northeast quadrant* and *counterclockwise for the northwest.* When the bearing is in the southerly quadrants, it's *counterclockwise to the east* and *clockwise to the west*, and we measure from South. One drawback to this system is that calculating the angles between bearings, or adding and subtracting them, is a source of mistakes when they are not fully understood.

Generally speaking, it is much easier to add and subtract angles when using ***azimuths***. An azimuth is the horizontal angle, measured clockwise, starting from North, to determine the direction of the line. The next thing to mention here is that both bearings and azimuths are in Degrees, Minutes and Seconds (D.ms) notation and the HP calculators do not add and subtract this way.

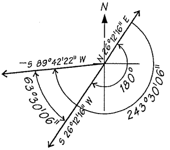
One solution is to turn the D.ms notation into decimal degrees (D.dd) before adding or subtracting them. After the calculation, change them back to D.ms. *Forgetting to change them, either initially or after the calculation is one of the most common causes of mistakes in surveying calculations.*



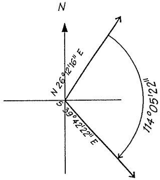
Directions by Azimuths



Adding and subtracting bearings can be confusing, depending on the quadrants they are in. Adding an angle to a bearing in the northeast quadrant, for instance, which makes the new bearing greater than 90° means that you have to convert it to a different quadrant.

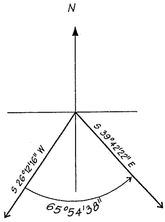


In the example to the right, adding 114°05'22" to the northeast bearing will give you an answer of 140°17'38", and put you into the southeast quadrant. To change it to a southeast bearing, you have to subtract 180° from it.



But, suppose the angle had taken you all of the way into the *southwest* quadrant?

The bearing angle of a line does not change, but the *direction* can. Going backwards on a northeast bearing takes you southwest . . . a 180° difference. When the angle exceeds that of a straight line (180°), subtract the 180° to make the calculation more manageable. By taking out the 180° portion and reversing the direction you also minimize mistakes in the calculation.



If you have a case where both the original bearing and the result are in the north quadrants (or both in the south), you simply subtract the original from the angle to add and add the angle to subtract. The example to the right shows both bearings in the southern quadrants.

65°54'38" - 26°12'16" = 39°42'22"

Major time is saved when working on the test problems if you do not have to do these calculations by hand. And, as mentioned before, the adding and subtracting of the angles and bearings are a major source of error. The best solution is to be able to let the calculator do the work. The first three programs we will put into the calculator will deal with changing azimuths to bearings, bearings to azimuths and just adding and subtracting in Degrees, minutes and seconds (D.ms).

First things first: Let's look at how the programs are designed to work. Chapter 12 of the manual that comes with the calculator should be read before continuing.

Some of the programs will use the INPUT function in the calculator. This built-in function will prompt for input with a single letter and a question mark, S? or T?, and in others we will build our own prompts to provide clarity when using a program. To do this, we input an equation that will act as a prompt when Flag 10 is set.

Setting (or clearing) Flag 10 is slightly different than the other flags. Use the keystrokes [right-shift] (right-shift) [×] (times) [1] (selects SF) [.] (inputs 1) [0] (adds the 0), the flag options are 1SF, 2CF, and 3FS?. Note that there is no test function for Flag Clear? to cause an action or branch, so in the programming this option is not available for branching.

Let's write a program . . . Remember, before starting you should **read chapter 12** in the user's manual that was furnished with your calculator, to get a general idea of how this is done.

This first program changes azimuths to bearings, and also contains the secondary Label that will allow us to access it from other programs without having the unneeded prompt portion. This is what allows us to offer the option of working in either azimuths or bearings during the coordinate geometry programs.

The actual steps for input of line A0003 were explained in detail on page 1. Start at the top of program memory by stroking [↰] [XEQ] [.] [.] and then stroke [↰] [R/S] to begin input of the program steps. Type in the program steps shown below.

A0001	LBL A	[↰] [+] [A]
A0002	SF 10	[↱] [×] [1] [.] [0]
A0003	AZIMUTH:	[↱] [STO] *then* [RCL] *before each alpha input*
N0001	LBL N	[↰] [+] [N]
N0002	→HR	[↰] [5]
N0003	ENTER	[ENTER]
N0004	ENTER	[ENTER]
N0005	90	[9] [0]
N0006	÷	[÷]
N0007	IP	[↱] [x^2]
N0008	1	[1]
N0009	+	[+]
N0010	STO Q	[STO] [Q]
N0011	R↓	[R↓]
N0012	ENTER	[ENTER]
N0013	SIN	[SIN]
N0014	ASIN	[↰] [SIN]
N0015	ABS	[↰] [y^x]
N0016	→HMS	[↱] [5]
N0017	STO B	[STO] [B]
N0018	RCL Q	[RCL] [Q]
N0019	RTN	[↱] [+]

You need to check that the program steps have been input correctly. If you stroke [↰] [x↔y], and then select PGM by stroking [2], you will see a list of the programs (at this point you only have two). As you scroll down the list of the program names you'll see the name and a 'LN' number. In this case

```
RPN
LBL A
LN=17
```

This number indicates the size of the program (in bytes). Hold down the [↱] key and press the [ENTER] key (SHOW) to see the check sum.

```
RPN
CK=B086
LN=17
```

Because we inserted a LBL N in the program as an entry point for other programming we will do, it is counted as a separate label by the calculator, so check it too. You should have LN=81 and CK=1DE5.

If you don't show the same numbers as those we've published it means something is wrong with the input. Go back and check your program steps for typos, extra (or missing) steps and make any necessary changes. Then check the size and check sum again. ***A complete chart of the LN and Checksums is on page 46.***

A couple of things to remember about this program when you are using it. Step N0002 changes your Degrees, Minutes and Seconds (D.ms) input into decimal degrees (D.dd) to use it for the calculations, and step N0016 changes it back to D.ms before displaying it. This means that your input and output will *always* be D.ms.

PROGRAM: AZIMUTH TO BEARING/QUADRANT CODE

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
		[XEQ] [A]	
AZIMUTH: 0.0000	Input the Azimuth (D.ms)	[R/S]	BEARING (D.ms) QUAD CODE
EXAMPLE: CHANGE THE AZIMUTH, 125°23'16", TO BEARING AND QUADRANT CODE		[XEQ] [A]	
AZIMUTH; 0.0000		[1] [2] [5] [.] [2] [3] [1] [6] [R/S]	54.3644 2.0000

Start at the top of program memory again by stroking [L-shift] [XEQ] [.] [.], stroke [L-shift] [R/S] to begin input of the program steps. Type in the new steps shown below. **Chapter 6** of the user's manual explains equation input. The [R↓] key is the colon, and the [R/S] key is the space. Stroke [RCL] before stroking the letter key for input of the alpha characters.

B0001 LBL B	[L-shift] [+] [B]
B0002 SF 10	[R-shift] [×] [1] [.] [0]
B0003 BEARING:	[R-shift] [STO] *then* [RCL] *before each alpha input*
B0004 STO B	[STO] [B]
B0005 QUAD CODE:	[R-shift] [STO] *then* [RCL] *before each alpha input*
B0006 STO Q	[STO] [Q]
B0007 x<>y	[x<>y]
B0008 →HR	[L-shift] [5]

Continued in next column

B0009 x<>y	[x<>y]
B0010 ENTER	[ENTER]
B0011 ENTER	[ENTER]
B0012 2	[2]
B0013 ÷	[÷]
B0014 IP	[R-shift] [x^2]
B0015 π	[R-shift] [COS]
B0016 →DEG	[L-shift] [6]
B0017 x	[×]
B0018 x<>y	[x<>y]
B0019 LASTx	[L-shift] [ENTER]
B0020 x	[×]
B0021 COS	[COS]
B0022 R↑	[R-shift] [R↓]
B0023 x	[×]
B0024 -	[-]
B0025 →HMS	[R-shift] [5]
B0026 RTN	[R-shift] [+]

LN=108 CK=B010

PROGRAM: BEARING/QUADRANT CODE TO AZIMUTH

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
		[XEQ] [B]	
BEARING:	Input the Bearing (D.ms)	[R/S]	
QUAD CODE:	Input the Quadrant Code	[R/S]	AZIMUTH (D.ms)

EXAMPLE: CHANGE THE BEARING, N 25°23'16" W, TO AN AZIMUTH		[XEQ] [B]	
PROMPT	**INSTRUCTIONS**	**KEYSTROKES**	**OUTPUT**
BEARING:	Input the Bearing (D.ms)	[2] [5] [·] [2] [3] [1] [6] [R/S]	
QUAD CODE:	Input the Quadrant Code	[4] [R/S]	334.3644

We'll add the short program (right) to our collection, (it adds and subtracts in D.ms) and then get some practice using all of the ones we've put in so far.

This one is different from the first two. In those, you execute the program and they prompt for the input. In this one, you input the numbers first and then execute the program. Again, start at the top of program memory by stroking [⬑] [XEQ] [·] [·], stroke [⬑] [R/S] to begin input of the program steps.

Type in the steps as shown below.

D0001	LBL D	[⬑] [+] [D]	Begin label A
D0002	x<>y	[x<>y]	Swap
D0003	→HR	[⬑] [5]	Convert to D.dd
D0004	x<>y	[x<>y]	Swap
D0005	→HR	[⬑] [5]	Convert to D.dd
D0006	x<>y	[x<>y]	Swap
D0007	+	[+]	Add
D0008	→HMS	[⬏] [5]	Convert to D.ms
D0009	RTN	[⬏] [+]	End

CK=0AA4
LN=27

PROGRAM: ADD OR SUBTRACT IN DEGREES, MINUTES OR SECONDS

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
	Input the 1st angle or azimuth (D.ms)	[ENTER]	
	Input the 2nd angle or azimuth (D.ms) (to subtract, stroke [+/-])	[XEQ] [D]	Sum or Difference
EXAMPLE: WHAT IS THE ANGLE BETWEEN N 17°22'41" W AND S 23°15'44" E?		[XEQ] [B]	
BEARING:		[2] [3] [·] [1] [5] [4] [4] [R/S]	
QUAD CODE:		[2] [R/S]	156.4416 (D.ms)
		[1] [7] [·] [2] [2] [4] [1] [XEQ] [D]	174.0657 (D.ms)

NOTE: The angle between S 23°15'44" E and N 17°22'41" W would be 185.5303 the answer is always the angle right in a clockwise direction from the first direction to the second direction

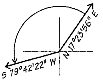
Exercise 1 (do the first two longhand, then complete the exercise with the programs):

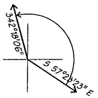
1. Add the angles, 28°15'34", 102°52'41", and 16°16'08" *ans*: ________________
2. Subtract 28°15'34 from 102°52'41", then add 16°16'08" *ans*: ________________
3. Add the angle, 102°52'41", to a bearing of N 62°45'23" W *ans*: ________________
4. Subtract 98°15'59" from a bearing of N 01°14'17" E *ans*: ________________

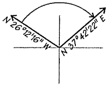
At this point, it's a safe assumption that you would rather use the hp33s to do this type of problem. You've also deleted two chances for error each time you use the programs. The 33s uses angle notation in the form of Degrees, decimal point, then the minutes and seconds. 62°45'23" would be input as 62.4523. The calculator also allows you to carry tenths or even hundredths of a second for more accuracy (Fix 5 or 6), but in surveying you decide what is acceptable precision.

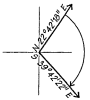
For the trig functions, use the calculator functions for changing the angles to and from decimal degrees (D.dd) before looking up the function, and then back to D.ms after looking up arcfunctions. The keystrokes to change from D.ms to D.dd are [left-shift] [5] and to do the reverse, [right-shift] [5].

Exercise 2 *(do the 1, 2, 5 and 6 longhand, then complete the exercise with the programs)*
Calculate the angles indicated



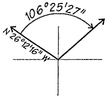


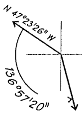


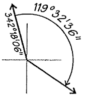


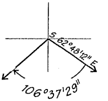
1. *ans:* __________ 2. *ans:* __________ 3. *ans:* __________ 4. *ans:* __________

Calculate the azimuth or bearing as indicated









5. *az:* __________ 6. *brg:* __________ 7. *brg:* __________ 8. *az:* __________

What are the answers to the following

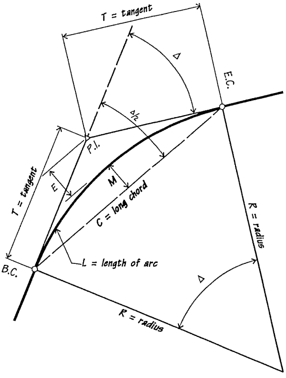
9. Cosine 17°15'23" __________ 10. Tangent 104°52'26" __________ 11. Sine 92°00'10" __________

-

12. Find the Sine of 197°14'23", then find the arcsine of the answer and change it back to D.ms. __________ Did you know that you just used this capability in a program, for converting quadrants?

Circular curves

You are probably very familiar with circular curves, but to better understand what these programs do, we'll review the definitions, symbols used, and the formulas for calculating the different parts.



Curve Nomenclature
The parts of a typical horizontal (circular) curve are shown above

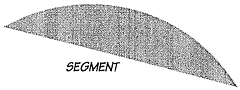
PI, Point of Intersection is the point where the two tangents intersect.

Central Angle, commonly called Delta (Δ) or I, it is the deflection angle measured, at the point of intersection, between the back and forward tangents. Most often this is one of the known parts essential to the calculation of other parts of the curve.

Deflection angle (ϕ), what we call the deflection angle is one-half the central angle, or the angle, turned at the BC of the curve, from the PI to the EC. There is also an angle from the BC to any point on the curve that is called the deflection angle when calculating stations for stakeout.

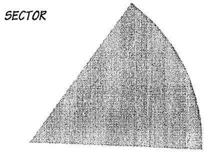
Length of curve (**L**), is the distance between the beginning and end of the curve measured along the curve. Arc Length.

$$L = 100\Delta R(\pi/180^{o})$$



Tangent Distance (**T**), actually a "semi-tangent" to the curve, it is the distance between the PI and the beginning or end of the curve and the two tangents are always equal.

$$T = R\tan\Delta/2$$

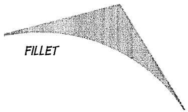


Radius (**R**), the "radius" is normally referred to by its length.

Chord (**LC**), the long chord is the distance between the beginning and end of the curve points of tangency, and is the length for angle Δ. Also called the 'chord', or 'short chord' is the distance from the BC of the curve to any point on the curve, for angle ϕ.

$$LC = 2R\sin\Delta/_2$$

$$C = 2R\sin\phi/_2$$



BC (or **PC**), the 'beginning of curve' or 'point of curvature' both common usage in different parts of the U.S.

EC (or **PT**) the 'end of curve' or 'point of tangency' both common usage in different parts of the U.S.

Middle Ordinate (**M**), length of ordinate from the middle of the long chord to the middle of the curve.

$$M = R(1-\cos\Delta/2) = R\,\text{vers}\,\Delta/2$$

External (**E**), distance from PI to the middle of the curve.

$$E = R(\sec\Delta/2-1) = R\,\text{exsec}\,\Delta/2$$

Sector area, the 'pie-shaped' area – from the radius point to the B.C., along the arc of the curve to the E.C. and back to the radius point.

$$\text{Sector area} = \pi R^2\Delta/360 = LR/2$$

Segment area, the area between the arc of the curve and the long chord of the curve.

$$\text{Segment area} = \text{Sector area} - \tfrac{1}{2}R^2\sin\Delta = \text{Sector area} - \tfrac{1}{2}CR\cos(\Delta/2)$$

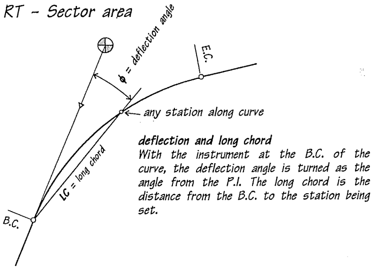
Fillet area, the area between the tangents of the curve and the arc of the curve. Often used for calculating pavement areas at the returns on street intersections.

$$\text{Fillet area} = RT - \text{Sector area}$$

Circular Curve Layout

The most common method for staking out a curve is the deflection-offset method, using long chord solutions to each of the station intervals to be staked.

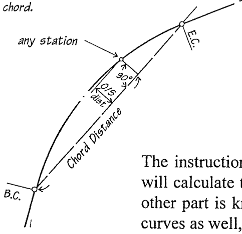
Once a curve has been calculated using the circular curve program, you can continue into the layout program to calculate the chord and deflection angles to any stations to be set.

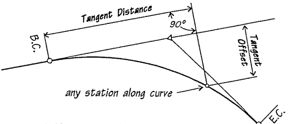


The layout program also calculates solutions for layout by the tangent-offset and chord-offset methods, and includes an option to stake the curve at an offset to the centerline instead of on the centerline itself. Offsets to the curve on the outside are input as positive, if the offset is to the inside of the curve, input the offset as negative.

Initial prompts are for selection of the type of output you want. Input a number for the type you want, and just stroke [R/S] for the others. This will be followed by a prompt for the offset, if any. To just calculate centerline stroke [R/S] or input the width of the offset and stroke [R/S]. The program will prompt for station and then output the stakeout information. You can run the program again to generate a different type, or input another curve and generate the stakeout information for it.

chord/offset method
Similar to the tangent/ofset method, except that the distance is pulled along the full chord of the curve to a point opposite the station being set. Often used on curves in tunnels with the laser line set on the long chord.





tangent/offset method
The tangent distance is pulled along the tangent line (semi-tangent of the curve), and a temporary point is set. The offset distance is measured to the station being set, at right angles to the tangent.

The instructions below start with the use of the circular curve program, which will calculate the curve if either the central angle or radius is known and if one other part is known. On the following pages we address the layout of circular curves as well, giving the user three methods of stakeout to chose from.

PROGRAM: CIRCULAR CURVES

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		Begin the program	[XEQ] [C]	
2	DELTA:	If the CENTRAL ANGLE is known, input it (D.ms format), if not known, no input **Either the central angle or radius *must* be input as one of the known parts**	[R/S]	
3	R?	If the RADIUS is known, input it, otherwise no input **Either the central angle or radius *must* be input as one of the known parts**	[R/S]	
4	L?	If known, input the length of arc, other-wise no input	[R/S]	
5	C?	If known, input the chord distance otherwise no input	[R/S]	
6	T?	If known, input the tangent distance otherwise no input	[R/S]	I= Central Angle
7				R= Radius
8			[R/S]	L= Arc Length
9			[R/S]	C= Chord Distance
10			[R/S]	

Continued on next page

PROGRAM: CIRCULAR CURVES (*Continued from previous page*)

11			[R/S]	T= Tangent Distance
12			[R/S]	E= External Distance
13			[R/S]	M= Mid-Ordinate Distance
14		AREAS: reminder displayed for 1 second	[R/S]	
15		SECTOR: reminder (1 second prompt)	[R/S]	SECTOR AREA
16		SEGMENT: reminder (1 second prompt)	[R/S]	SEGMENT AREA
17		FILLET: reminder (1 second prompt)	[R/S]	FILLET AREA
18	MORE=0 STAKE=1	To calculate another curve, input 0, **or** to calculate stakeout for this curve input 1 **Or**, to leave the program stroke [C]	[R/S]	

Exercise 3:

Complete the curve data for the following:

1.

Radius = 510.23'

Delta = ______

Length = ______

Tangent = ______

Chord = 244.77'

External = ______

Mid-Ordinate = ______

2.

Radius = 400.00'

Delta = ______

Length = ______

Tangent = 125.16'

Chord = ______

External = ______

Mid-Ordinate = ______

Sector = ______

Segment = ______

Fillet = ______

3.

Radius = 200.00'

Delta = ______

Length = 10.26'

Tangent = ______

Chord = ______

External = ______

Mid-Ordinate = ______

Sector = ______

Segment = ______

Fillet = ______

4.

Radius = ______

Delta = 1°25'16"

Length = ______

Tangent = ______

Chord = 400.00'

External = ______

Mid-Ordinate = ______

PROGRAM: CIRCULAR CURVE LAYOUT

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		After calculation of the curve, begin the curve layout program by responding [1] to the final prompt in the curve program		
2	SELECT TYPE	Reminder prompt will be displayed for one second		
3	DEFLECTION	To select this option, input any number and stroke [R/S] **OR** to not use this type, just stroke [R/S] without input		
4	TAN-OS	To select this option, input any number and stroke [R/S] **OR** to not use this type, just stroke [R/S] without input		
5	CHD-OS	To select this option, input any number and stroke [R/S] **OR** to not use this type, just stroke [R/S] without input		
6	OFFSET=:	To calculate the layout at an offset, input the size of the offset (change sign for offsets to the inside of the curve). For calculations along centerline, just stroke [R/S] without input	[R/S]	
7	BC STATION:	Input the station of the B.C.	[R/S]	
8	INPUT STATION:	Input the station you want to calculate layout for. Output is selection dependent	[R/S]	
9	DEF ANGLE TAN DIST or CHD DIST	Selection dependent prompt	[R/S]	
				VALUE
			[R/S]	
10	CHORD or OFFSET	Selection dependent prompt	[R/S]	
11				VALUE
12			[R/S]	
	INPUT STATION:	Returns you to step 8. Input the next station you want to calculate and repeat steps 8 through 12, or stroke [C] to leave the layout program		

Exercise 4:
Complete the curve data for the following, and then calculate the layout information for the stations as indicated:

1. Layout by deflection and chord

		STATION	DEFLECTION	CHORD
Radius =	510.23'	12+19.23 B.C.		
Delta =		12+50		
Length =		13+00		
Tangent =		13+50		
Chord =	244.77'	14+00		
External =		14+50		
Mid-Ordinate =		E.C.		

2. Layout by tangent-offset

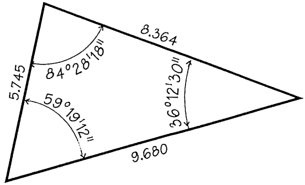
		STATION	TANGENT DIST.	OFFSET
Radius =	400.00'	122+34.97		
Delta =		122+50		
Length =		123+00		
Tangent =	125.16'	123+50		
Chord =		124+00		
External =		124+50		
Mid-Ordinate =		E.C.		

3. Layout by chord-offset

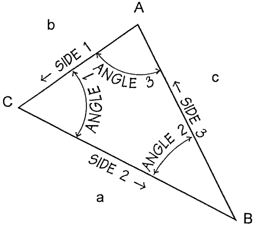
		STATION	CHORD DIST.	OFFSET
Radius =		53+24.37		
Delta =	35°15'22"	53+50		
Length =	237.71'	54+00		
Tangent =		54+50		
Chord =		55+00		
External =		55+50		
Mid-Ordinate =		E.C.		

triangle solutions

The triangle shown to the right will be used for the examples. It should be noted that the output will vary slightly, depending on the number of places input, particularly in the input of the angles.

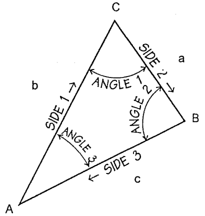


The notations for angles and sides is familiar to HP users, but is not the standard, or *textbook* notation which you will have learned in trigonometry (side **a** opposite angle **A**, side **b** opposite angle **B**, and side **c** opposite angle **C**). The sides and angles are numbered, in order, going around the triangle as shown below.



In the example shown below, the assigned designations go clockwise, if it will better fit the information available, the labeling may go counter-clockwise instead, as shown to the left.

Remember, side 1 is wherever you put it.



The example triangle (above) shows this style of labeling, compared to the standard notation for sides and angles. ***Side 1*** may be assigned to any side that is convenient to use, depending upon the available information about the triangle. It should be located at a side where the known information then falls into position for solution by one of the routines.

The program begins with selection prompts for you to select the TYPE of solution needed. A brief reminder prompt, TYPE SELECTION, is displayed (one second) and then the 5 type options are offered. To ***NOT*** select one of the types just stroke [R/S]. To select one, input 1 (or any number, the [7] key is right next to [R/S] and is easy to stroke). Continue with [R/S] through any remaining type options until you get the first of the three input prompts.

The input prompts shown will depend on your selected solution type.

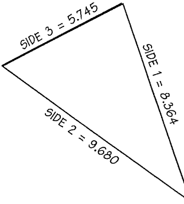
NOTE! There is no solution for a triangle where the three angles are the only known parts, since this condition can produce an infinite number of similar triangles.

PROGRAM: TRIANGLE SOLUTIONS

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		Begin the program	[XEQ] [G]	
2	TYPE SELECTION	APPEARS BRIEFLY AS A REMINDER		
3	S1-S2-S3	SOLUTION FOR THREE SIDES KNOWN	Input a number to select this option or [R/S] to continue to next prompt	
4	A3-S1-A1	SOLUTION FOR TWO ANGLES AND THE INCLUDED SIDE KNOWN	Input a number to select this option or [R/S] to continue to next prompt	
5	S1-A1-A2	KNOWN SIDE AND THE NEXT TWO FOLLOWING ANGLES KNOWN	Input a number to select this option or [R/S] to continue to next prompt	
6	S1-A1-S2	SOLUTION WHEN TWO SIDES AND THE INCLUDED ANGLE ARE KNOWN	Input a number to select this option or [R/S] to continue to next prompt	
7	S1-S2-A2	TWO SIDES AND THE FOLLOWING ANGLE ARE KNOWN	Input a number to select this option or [R/S] to continue to next prompt	
8		Three input prompts will be shown. Input the value requested	[R/S] after each input	
9				SIDE 1
10			[R/S]	ANGLE 1
11			[R/S]	SIDE 2
12			[R/S]	ANGLE 2
13			[R/S]	SIDE 3
14			[R/S]	ANGLE 3
15			[R/S]	AREA
16	2ND SOLUTION	Appears when there is a second solution to the S1-S2-A2 type. The program will again repeat steps 9 through 15 but the output will not be in the same order as the original input	[R/S]	
17		When finished with the calculations, it will return you to the triangle program for next calculation, beginning at step 2	[R/S]	

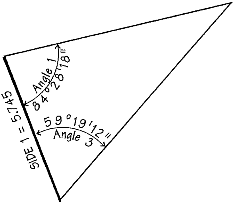
Examples are shown on the following pages for each of the solution types individually, covering the prompt, input and output in each case.

	PROMPT	KEYSTROKES	OUTPUT
8a	INPUT SIDE 1	[8][.][3][6][4][R/S]	
8b	INPUT SIDE 2	[9][.][6][8][R/S]	
8c	INPUT SIDE 3	[5][.][7][4][5][R/S]	
9			SIDE 1
		[R/S]	8.3640
10		[R/S]	ANGLE 1
		[R/S]	36.1232
11		[R/S]	SIDE 2
		[R/S]	9.6800
12		[R/S]	ANGLE 2
		[R/S]	59.1912
13		[R/S]	SIDE 3
		[R/S]	5.7450
14		[R/S]	ANGLE 3
		[R/S]	84.2816
15		[R/S]	AREA
		[R/S]	23.9138



side 1, side 2, side 3

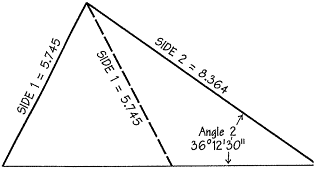
THREE SIDES KNOWN is one of the most used solutions for triangles, particularly since the accuracy of electronic distance measurement trilateration has, for the most part, replaced triangulation in several types of surveys. The example begins at the input prompts.



angle 3, side 1, angle 1

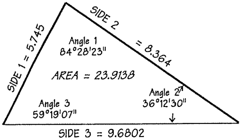
TWO ANGLES AND THE INCLUDED SIDE ARE KNOWN This solution is also used as a secondary solution to some of the other routines, after the problem has first been reduced to these three known parts.

	PROMPT	KEYSTROKES	OUTPUT
8a	INPUT ANGLE 3	[5][9][.][1][9][1][2] [R/S]	
8b	INPUT SIDE 1	[5][.][7][4][5][R/S]	
8c	INPUT ANGLE 1	[8][4][.][2][8][1][8] [R/S]	
9			SIDE 1
		[R/S]	5.7450
10		[R/S]	ANGLE 1
		[R/S]	84.2818
11		[R/S]	SIDE 2
		[R/S]	8.3641
12		[R/S]	ANGLE 2
		[R/S]	36.1230
13		[R/S]	SIDE 3
		[R/S]	9.6801
14		[R/S]	ANGLE 3
		[R/S]	59.1912
15		[R/S]	AREA
		[R/S]	23.9142

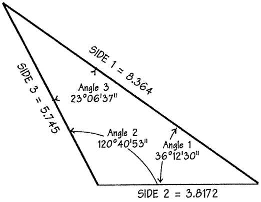


side 1, side 2, angle 2

TWO SIDES AND THE FOLLOWING ANGLE KNOWN has two possible solutions. When this configuration is used , both solutions may be output. The second solution will not necessarily show the parts in the same order as the input.



First Solution Output



Second Solution Output

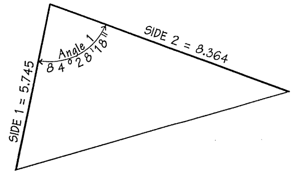
	PROMPT	KEYSTROKES	OUTPUT
8a	INPUT SIDE 1	[5] [·] [7] [4] [5] [R/S]	
8b	INPUT SIDE 2	[8] [·] [3] [6] [4] [R/S]	
8c	INPUT ANGLE 2	[3] [6] [·] [1] [2] [3] [R/S]	
9			SIDE 1
		[R/S]	5.7450
10		[R/S]	ANGLE 1
		[R/S]	84.2823
11		[R/S]	SIDE 2
		[R/S]	8.3640
12		[R/S]	ANGLE 2
		[R/S]	36.1230
13		[R/S]	SIDE 3
		[R/S]	9.6802
14		[R/S]	ANGLE 3
		[R/S]	59.1907
15		[R/S]	AREA
		[R/S]	23.9139
		[R/S]	
	2ND SOLUTION	[R/S] (output will continue with the second solution option)	
16			SIDE 1
		[R/S]	8.3640
17		[R/S]	ANGLE 1
		[R/S]	36.1230
18		[R/S]	SIDE 2
		[R/S]	3.8172
19		[R/S]	ANGLE 2
		[R/S]	120.4053
20		[R/S]	SIDE 3
		[R/S]	5.745
21		[R/S]	ANGLE 3
		[R/S]	23.0637
22		[R/S]	AREA
		[R/S]	9.4301

Note that the output is not in the same order as the original input.

	PROMPT	KEYSTROKES	OUTPUT
8a	INPUT SIDE 1	[5][·][7][4][5][R/S]	
8b	INPUT ANGLE 1	[8][4][·][2][8][1][8] [R/S]	
8c	INPUT SIDE 2	[8][·][3][6][4][R/S]	
9			SIDE 1
		[R/S]	5.740
10		[R/S]	ANGLE 1
		[R/S]	84.2818
11		[R/S]	SIDE 2
		[R/S]	8.3640
12		[R/S]	ANGLE 2
		[R/S]	36.1231
13		[R/S]	SIDE 3
		[R/S]	9.6800
14		[R/S]	ANGLE 3
		[R/S]	59.1911
15		[R/S]	AREA
		[R/S]	23.9138

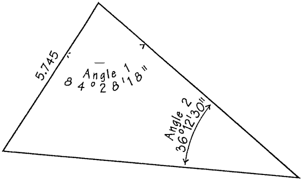
side 1, angle 1, side 2

TWO SIDES AND THE INCLUDED ANGLE KNOWN is resolved by finding the third side, and then solving the triangle as shown on the previous page, three sides known.



side 1, angle 1, angle 2

ONE SIDE AND THE TWO FOLLOWING ANGLES KNOWN This solution first solves for the third angle. The remainder of the triangle is solved as Angle, Side, Angle to determine the other missing sides.



	PROMPT	KEYSTROKES	OUTPUT
8a	INPUT SIDE 1	[5][·][7][4][5][R/S]	
8b	INPUT ANGLE 1	[8][4][·][2][8][1][8] [R/S]	
8c	INPUT ANGLE 2	[3][6][·][1][2][3][R/S]	
9			SIDE 1
		[R/S]	5.7450
10		[R/S]	ANGLE 1
		[R/S]	84.2818
11		[R/S]	SIDE 2
		[R/S]	8.3641
12		[R/S]	ANGLE 2
		[R/S]	36.1230
13		[R/S]	SIDE 3
		[R/S]	9.6801
14		[R/S]	ANGLE 3
		[R/S]	59.1912
15		[R/S]	AREA
		[R/S]	23.9142

Exercise 5:

1. Solve the triangle

Side 1 ____________________

Angle 1 ____________________

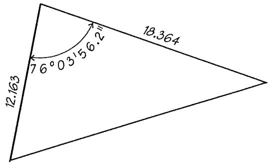
Side 2 ____________________

Angle 2 ____________________

Side 3 ____________________

Angle 3 ____________________

Area ____________________



2. Solve the triangle

Side 1 ____________________

Angle 1 ____________________

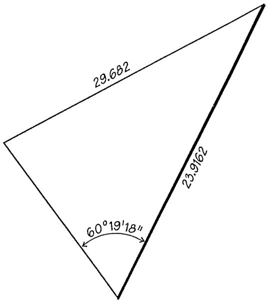
Side 2 ____________________

Angle 2 ____________________

Side 3 ____________________

Angle 3 ____________________

Area ____________________



3. Solve the triangle

Side 1 ____________________

Angle 1 ____________________

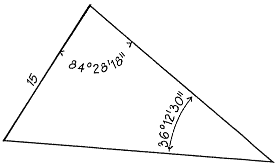
Side 2 ____________________

Angle 2 ____________________

Side 3 ____________________

Angle 3 ____________________

Area ____________________



Exercise 5 *(Cont'd):*

4. Solve the triangle

Side 1 ____________

Angle 1 ____________

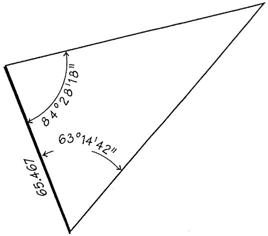
Side 2 ____________

Angle 2 ____________

Side 3 ____________

Angle 3 ____________

Area ____________



52. Solve the triangle

Side 1 ____________

Angle 1 ____________

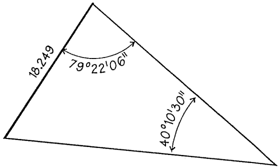
Side 2 ____________

Angle 2 ____________

Side 3 ____________

Angle 3 ____________

Area ____________



6. Solve the triangle

Side 1 ____________

Angle 1 ____________

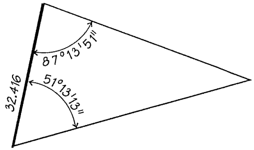
Side 2 ____________

Angle 2 ____________

Side 3 ____________

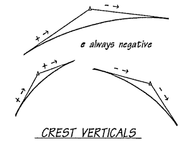
Angle 3 ____________

Area ____________



vertical curves and grades

Vertical curves are usually described as 'crest' or 'sag' verticals, as shown to the right.



The form of the curve may be expressed as

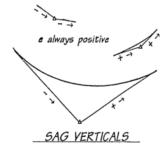
$$y = ax^2 + bx$$

where y is the height of the curve above or below the first tangent point, and x is the distance therefrom.

The **highest** or **lowest** point on the curve is at a point where the gradient of the tangent is equal to **0%**. This is called the "turning point" of the curve. If both gradients have the same sign, there is no actual turning point, and the vertical direction is continuous. The gradient of the tangent may be found by differentiating y with respect to x in the equation above.

When

$$dy/dx = 0, \quad x = -b/2a.$$



Our program for vertical curves is designed to do quick vertical curve and grade calculations. The number of entries you make during input tells the program whether you are calculating a grade or a curve.

Formulas for vertical curves and grades vary with the known values when you begin to solve the grade or curve. In most vertical curve cases, if you are working from a set of plans, you would know the beginning station (BVC) and it's elevation, the ending station (EVC) and it's elevation, the length and the grade in (G*i*) and the grade out (G*o*). In most cases, the intersection point (PVI) is given too. The following would apply:

If the high or low point elevation and the beginning station (EL_0 and PVC) are known,

1. $$\left(\frac{Go - Gi}{200L}\right)(STA - BVC)^2 + \left(\frac{Gi}{100}\right)(STA - BVC) + (El_{bvc} - El_{sta}) = 0 \qquad (ax^2 + bx + c = 0)$$

If the high or low point elevation and the intersection station (EL_0 and PVI) are known,

2. $$L = 200(El_{bvc} - El_0)(Go - Gi)\left(\frac{1}{Gi^2}\right)$$

Where:

Gi = Beginning grade (grade in), expressed in percent
Go = Ending grade (grade out), expressed in percent
L = Length of curve, measured in along the horizontal
STA = Station along horizontal with curve elevation
El_{sta} = elevation at STA
BVC = Beginning station (point of curve)
El_{bvc} = Beginning elevation at BVC
PVI = Point of tangent intersection
El_{pvi} = Elevation at the PVI
EL_0 = Elevation at high or Low point of curve
EVC = Ending station (end of curve)
El_{evc} = Elevation at the EVC

There is a question that often occurs in tests, but is never used in the real world; Given the High/Low point elevation, the grades in and out, and either the PVI elevation or the BVC elevation and want to know the minimum length of curve that will work. Minimum lengths are NEVER used, and the difference in grades is used to select the required length from a table that takes passing sight distance into consideration.

If the PVI is given

$$L = 200(EL_{pvi} - EL_0)(G_0 - G_i)\left(\frac{1}{G_i^2}\right)$$

or, if the BVC is given.

$$L = 200(EL_{bvc} - EL_0)(G_0 - G_i)\left(\frac{1}{G_0 G_i}\right)$$

We've not included a program that will do this type of problem but do suggest that the above formulas can be input as equations in the equation library in case you need them. rather than include a program for this one case, we programmed for the day to day vertical calculations that you are more likely to encounter.

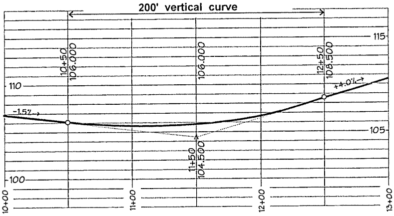
PROGRAM: CALCULATING ALONG A VERTICAL TANGENT OR CURVE

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
		[XEQ] [V]	
INPUT BEG STA	Input the station at the B.V.C.	[R/S]	
INPUT BEG ELEV	Input the elevation at the B.V.C.	[R/S]	
GRADE IN:	Input the % of grade for the tangent. (For a curve, input the % of grade for the incoming grade) Change sign if negative	[R/S]	
GRADE OUT:	**No input for a vertical tangent.** For a curve, input the % of grade for the outgoing grade) Change sign if negative	[R/S]	
INPUT LENGTH	**No input for a vertical tangent.** (For a curve, input the length of the vertical curve)	[R/S]	
	When calculating along a vertical curve, the turning point station and elevation are automatically output at this point. Stroke [R/S] to continue		STATION @ 0% ELEVATION @ 0%
INPUT STA:	Input the next station you want to calculate the elevation for	[R/S]	STATION ELEVATION
	After writing down the answers, stroke [R/S] to continue with the next station	[R/S]	
INPUT STA:	When finished with the calculations	[C]	

EXAMPLE

The vertical curve shown to the right will be used for the example. The B.V.C. station is 10+50, at elevation 106.00.

We will calculate the elevations for stations at 50 foot intervals along the curve, the 0% station and elevation (high/low point), in this example the low point. You can also calculate the station at which a particular elevation occurs, using [LBL] [E], after all of the information has been entered for the vertical tangent or curve you are working on.



PROGRAM: CALCULATING ALONG A VERTICAL TANGENT OR CURVE

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
		[XEQ] [V]	
INPUT BEG STA	Input the station at the B.V.C.	[1] [0] [5] [0] [R/S]	
INPUT BEG ELEV	Input the elevation at the B.V.C.	[1] [0] [6] [R/S]	
GRADE IN:	Input the % of grade for the incoming grade. (Change sign if negative)	[1] [.] [5] [+/-] [R/S]	
GRADE OUT:	Input the % of grade for the outgoing grade Change sign if negative	[4] [R/S]	
INPUT LENGTH	Input the length of the vertical curve. **Output is high/low point sta & elev**	[2] [0] [0] [R/S]	1104.5455 105.5909
		[R/S]	
INPUT STA:	Input the next station	[1] [1] [0] [0] [R/S]	1100.0000 105.5938
		[R/S]	
INPUT STA:	Input the next station	[1] [1] [5] [0] [R/S]	1150.000 105.8750
		[R/S]	
INPUT STA:	Input the next station	[1] [2] [0] [0] [R/S]	1200.000 106.8438
		[R/S]	
INPUT STA:	Input the next station	[1] [2] [5] [0] [R/S]	1250.000 108.5000
		[C]	

PROGRAM: CALCULATE STATION WHEN ELEVATION IS KNOWN

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
None . . .	Input the elevation (two stations will be shown, verify that they are in the curve to be valid answers.	[XEQ] [E]	STATION STATION
	Next output is the high/low point	[R/S]	STATION ELEVATION
	Return to original program	[R/S]	
EXAMPLE: FIND THE STATION AT WHICH ELEVATION 105.65 OCCURS		[1] [0] [5] [·] [6] [5] [XEQ] [E]	
	Output is station(s) at which the elevation occurs.		1125.2800 1083.8100 [R/S]
	Output is the high/low point		1104.5455 105.5909 [R/S]
INPUT STA:	Continue with input in main program		

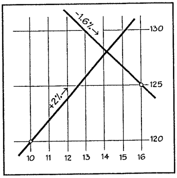
There are also times when you have known stations and elevations along two vertical tangents and need to calculate the point of intersection between them. This program will calculate the intersection point when the grades, a starting point, and an ending point are known.

Once the point of intersection (P.V.I.) is known, a curve length may be selected and a B.V.C. station and elevation calculated. From there, use the vertical program to calculate the stations along the curve. The program is accessed by stroking [XEQ] [P].

PROGRAM: CALCULATE VERTICAL INTERSECTION

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
		[XEQ] [P]	
INPUT STA 1	Input the first station	[R/S]	
INPUT ELEV 1	Input the elevation	[R/S]	
INPUT STA 2	Input the second station	[R/S]	
INPUT ELEV 2	Input the elevation	[R/S]	
GRADE IN:	Input the % of grade for the grade in. (Change sign if negative)	[R/S]	
GRADE OUT:	Input the % of grade for the grade out (Change sign if negative)	[R/S]	PVI STATION ELEVATION

The example will use the information from the illustration to the right. The known station at the beginning is 10+00, with an elevation of 120.00 and the known station at the end is 16+00 at elevation 125.00.



The two known grades are a +2.00 percent and a minus 1.60 percent. Follow the procedure below to obtain the station and elevation of the point of intersection.

EXAMPLE: FIND THE STATION AND ELEVATION OF THE VERTICAL INTERSECTION

PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
		[XEQ] [P]	
INPUT STA 1	Input the first station	[1] [0] [0] [0] [R/S]	
INPUT ELEV 1	Input the elevation	[1] [2] [0] [R/S]	
INPUT STA 2	Input the second station	[1] [6] [0] [0] [R/S]	
INPUT ELEV 2	Input the elevation	[1] [2] [5] [R/S]	
GRADE IN:	Input the % of grade for the grade in (Change sign if negative)	[2] [R/S]	
GRADE OUT:	Input the % of grade for the grade out (Change sign if negative)	[1] [.] [6] [+/-] [R/S]	1405.5556 128.1111

Exercise 6:

1. Using the information from the example above, calculate a 400' vertical curve to be used to round the grade along the roadway. Calculate the following:

BVC station ________________ elevation ____________ high point station ____________

EVC station ________________ elevation ____________ high point elevation __________

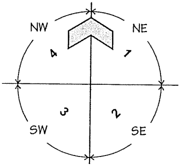
2. Calculate the elevations for the following stations:

12+00 ________ 12+50 ________ 13+00 ________ 13+50 ________ 14+00 ________

14+50 ________ 15+00 ________ 15+50 ________ 16+00 ________

3. At what stations will the elevations 123.58 and 121.56 occur? _________ & _________

coordinate geometry



The backbone of Coordinate Geometry calculations is the traverse program. We've tried to make this one as flexible as possible, interfacing with our inverse and closure programs.

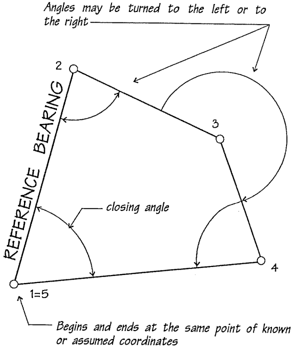
Bearings are input with quadrant codes and the quadrants are numbered with the same system that has been used by Hewlett-Packard since the first surveying programs for handheld HPs came out.

The bearing and quadrant code are prompted for, and all input (and output) is in Degrees,minutes and seconds (D.ms) format. After each input the R/S key is stroked.

A traverse may be thought of as either a "*closed*" or an "*open*" traverse. For use with this program, a **CLOSED traverse** may be either of **two** types. What we will call the **Type A** is similar to the one shown to the right.

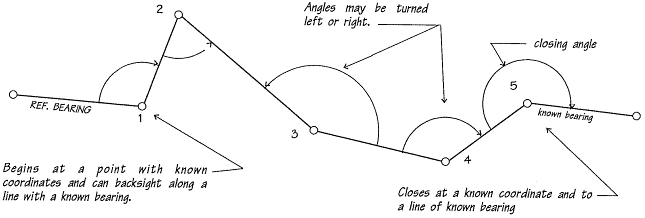
In this type of traverse, the line from #1 to #2 is usually a known line which is included in the traverse. The two points used would be part of a property or monument line, and the basis of bearings would be the bearing of the line.

This type of traverse also closes back to the original point of beginning, and allows the turning of a closing angle, which is turned at the first (and last) point, foresighting the second point.



Type 'A' Closed Traverse

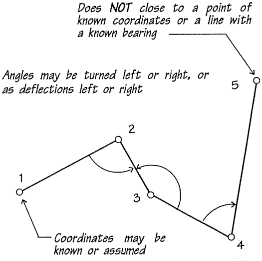
Type 'B' Closed Traverse



What we will consider to be a **Type B** closed traverse is one which begins at one known point and ends at another known point. For this type (previous page) the basis of bearings is usually obtained by backsighting another known point.

An *OPEN TRAVERSE* is one which, while it may begin at a known point, does **NOT** close to any point or line which allows adjustment of the traverse.

An *OPEN TRAVERSE* may also be considered as being an 'unfinished' traverse, in that it could later be used as a portion of a traverse which will be closed.



An Open Traverse

While an open traverse can not be closed, the Type 'B' closed traverse can. When you get to the end of the traverse, you can use the interfacing features of the **inversing program** *(page 29)* to calculate the distance and bearing from your end point to the coordinates of the point it is trying to match. A closure routine is essentially useless in this case, since it does not enclose an area (even though one is calculated). The precision, in this case, is the total distance traversed divided by the length of the error of closure.

Often the NCEES test questions will require you to calculate a small traverse before you can answer the question, but the question will be to determine the precision of the traverse or the area the traverse encloses.

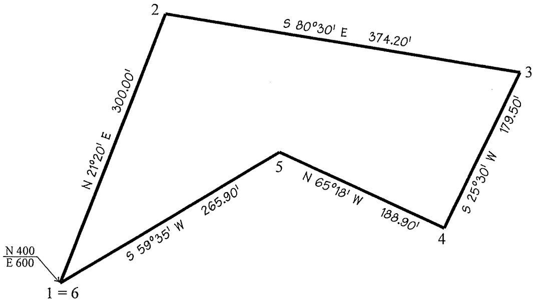
The precision of a closed traverse can be calculated by dividing the sum of the distances by the distance of the closure error. We have included a 'closure' routine, LBL K, which may be used immediately after running the traverse calculations and will complete the needed information. It calculates the error of closure, area and precision.

The programming instructions begin on the next page, and we have written them to include the use of the closure routine as a part of doing the traverse. Following that, we have proceeded to the inversing program.

That program has been made flexible enough to interface with the traverse, do individual inverses or to be used for inversing layout ties. We recommend that you read through this whole group before beginning to input the programs. Then take a break on the program input before tackling the intersections.

PROGRAM: TRAVERSE and CLOSURE

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		Begin the traverse program	[XEQ] [T]	
2	AZ=0 BRG=1	**Selection Prompt** To work in Azimuth mode, input 0. To work in Bearing/Quad mode input 1.	[0] or [1] then [R/S]	
3	NORTHING:	Input the North coordinate of the beginning point.	[R/S]	
4	EASTING:	Input the East coordinate of the beginning point.	[R/S]	
5	AZIMUTH:	**If Azimuth mode was selected** Input the azimuth that defines the direction of the first course.	[R/S]	
	OR			
5a	BEARING:	**If Bearing/Quad mode was selected (a)** Input the value of the bearing that defines the direction of the first course.	[R/S]	
5b	QUAD CODE:	**If Bearing/Quad mode was selected (b)** Input the quadrant code for the bearing that was just input.	[R/S]	
	THEN			
6	DISTANCE:	Input the distance for this course.	[R/S]	N= NNNNN.NNNN
			[R/S]	E= EEEEE.EEEE
			[R/S]	
		Returns to the prompt for the direction of the next course (steps 5 & 6 above)		
		Continue through the remaining courses of the traverse, repeating steps 5 and 6, until you have finished the calculations of the coordinates		
7		**Begin the closure output**	[XEQ] [K]	
8		CLOSE ERROR reminder prompt	[R/S]	DIRECTION DISTANCE
9		PRCSN 1: reminder prompt	[R/S]	PRECISION RATIO
10		AREA: reminder prompt	[R/S]	AREA
11		SUM H DIST: reminder prompt	[R/S]	PERIMETER DIST.
		When finished with the calculations	[C]	



EXAMPLE / EXERCISE: Calculate the coordinates for points 2 through 6 in the figure above, then calculate the error of closure, precision and area.

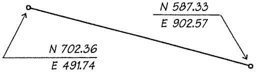
PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
	Begin the traverse program	[XEQ] [T]	
AZ=0 BRG=1	**Selection Prompt** The example is in Bearing/Quad so it's easier to work in that mode	[1] [R/S]	
NORTHING:	Input the North coordinate of the beginning point.	[4][0][0] [R/S]	
EASTING:	Input the East coordinate of the beginning point.	[6][0][0] [R/S]	
BEARING:	Input the value of the bearing that defines the direction of the first course.	[2][1][·][2][0] [R/S]	
QUAD CODE:	Input the quadrant code for the bearing that was just input.	[1] [R/S]	
DISTANCE:	Input the distance for this course.	[3][0][0] [R/S]	N= 679.4439
		[R/S]	E= 709.1380
		[R/S]	
BEARING:	**Continue with the traverse, noting the coordinates as you go. When finished, calculate the closure**	[XEQ] [K]	

PROGRAM: INVERSING – USED SEPARATELY

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		be sure to clear flag 4 before starting	[■] [×] [2] [4] [XEQ] [L]	
2	AZ=0 BRG=1	**Selection Prompt** Work in whatever mode is needed	[0] or [1] then [R/S]	
3	NORTHING:	Input the North coordinate of the beginning point.	[R/S]	
4	EASTING:	Input the East coordinate of the beginning point.	[R/S]	
5	NORTHING:	Input the North coordinate of the ending point.	[R/S]	
6	EASTING:	Input the East coordinate of the ending point.	[R/S]	
7		**If Azimuth mode was selected**	[R/S]	AZIMUTH
	OR			
7a		**If Bearing/Quad mode was selected**	[R/S]	BEARING
7b		**If Bearing/Quad mode was selected**	[R/S]	QUAD CODE
8			[R/S]	DISTANCE
9	MORE=0 STAKE=1	**Selection Prompt** Work another or radial stake points	[0] or [1] [R/S] or [C] to leave program	

EXAMPLE: INVERSING – USED SEPARATELY

Calculate the azimuth and bearing between the two points shown to the right.



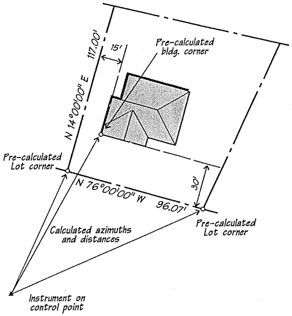
	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		be sure to clear flag 4 before starting	[■] [×] [2] [4] [XEQ] [L]	
2	AZ=0 BRG=1	Select azimuth output	[0] [R/S]	
3	NORTHING:	Input the North coordinate of the beginning point.	[7] [0] [2] [·] [3] [6] [R/S]	
4	EASTING:	Input the East coordinate of the beginning point.	[4] [9] [1] [·] [7] [4] [R/S]	
5	NORTHING:	Input the North coordinate of the ending point.	[5] [8] [7] [·] [3] [3] [R/S]	
6	EASTING:	Input the East coordinate of the ending point.	[9] [0] [2] [·] [5] [7] [R/S]	A= 105.3831
			[R/S]	D= 426.6300
	MORE=0 STAKE=1	**Selection Prompt** Work another or radial stake points	[1] or [0] [R/S] or [C] to leave program	Returns to prompts for next input

You can also use this program to inverse during a traverse. The result is not saved, and does not affect the traverse in progress. If you have the coordinates of the area you are surveying and are running a random traverse toward the next corner, inversing to the corner at each setup point will keep you going in the right direction. **This use of the program outputs the direction from the current traverse point.**

PROGRAM: INVERSING – USED DURING A TRAVERSE

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		To do an inverse during a traverse, set Flag 4 before beginning	[↱] [×] [1] [4] [XEQ] [L]	
2	AZ=0 BRG=1	**Selection Prompt** Select the same mode that you are using for the traverse	[0] or [1] then [R/S]	
3	NORTHING:	Input the North coordinate of the ending point.	[R/S]	
4	EASTING:	Input the East coordinate of the ending point.	[R/S]	
5		**If Azimuth mode was selected**	[R/S]	AZIMUTH
	OR			
5a		**If Bearing/Quad mode was selected**	[R/S]	BEARING
5b		**If Bearing/Quad mode was selected**	[R/S]	QUAD CODE
6			[R/S]	DISTANCE
7		Returns you to the traverse program	[R/S]	Next direction prompt

And, you can use the inverse program for stakeout. If you have pre-calculated coordinates for the points you want to stake, as shown to the right. A general procedure would be to inverse from the instrument point to the coordinates of the backsight, to get the azimuth to enter into the instrument while sighting the backsight and then select stake at the MORE=0 STAKE=1 selection prompt.



The program will now inverse to each of the points for staking azimuth and distance as you input the coordinates for the next point. Note that the MORE=0 STAKE=1 prompt appears at the end of each calculation, so you can continue normal inverses as well as stake your points.

Printing Error Note: On Page P12, the line numbering for LBL L, after step L0022 will change after the next step (LBL R) is typed in and will be R0022, R0023, R0024 . . . R0058 for the rest of the program, instead of L#### as printed in the program pages.

PROGRAM: INVERSING – USED FOR STAKEOUT

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		be sure to clear flag 4 before starting	[↱] [×] [2] [4] [XEQ] [L]	
2	AZ=0 BRG=1	**Selection Prompt** Work in whatever mode is needed	[0] or [1] then [R/S]	
3	NORTHING:	Input the North coordinate of the beginning point.	[R/S]	
4	EASTING:	Input the East coordinate of the beginning point.	[R/S]	
5	NORTHING:	Input the North coordinate of the ending point.	[R/S]	
6	EASTING:	Input the East coordinate of the ending point.	[R/S]	
7		**If Azimuth mode was selected**	[R/S]	AZIMUTH
	OR			
7a		**If Bearing/Quad mode was selected**	[R/S]	BEARING
7b		**If Bearing/Quad mode was selected**	[R/S]	QUAD CODE
8			[R/S]	DISTANCE
9	MORE=0 STAKE=1	**Selection Prompt** Work another or radial stake points	[1] [R/S]	
10	NORTHING:	Input the North coordinate of the next point to be staked	[R/S]	
11	EASTING:	Input the East coordinate of the next point to be staked	[R/S]	
12	MORE=0 STAKE=1	**Selection Prompt** Work another or radial stake points	[0] or [1] [R/S] or [C] to leave program	

Exercise 7:

Use the inverse program to calculate the stakeout ties to the lot and building corners in the figure on the opposite page if the coordinates of the control point are N 500.00 and E 1100.00. The coordinates of the lot corners are: Southwest corner N 527.9300, E 1214.0600; Southeast corner N 504.6886, E 1307.2763 and the building corner coordinates are N 553.4100, E 1235.8721.

To the southwest Lot Corner ______________________

To the southeast Lot Corner ______________________

To the building corner ______________________

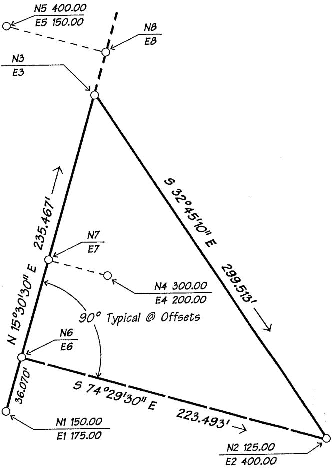
intersection solutions

The solutions to intersections problems are needed all of the time in surveying. We use an intersection formula to find out where two lines cross, then make that point the new IP or lot corner. Or, we need to know how far a point is offset from a given line, and at what distance from the line's origin. Next to just plain traversing, this is the most used type of calculation in surveying.

We've included all of the normal solutions, the type being chosen as part of the input and then worked by use of flags instead of using up four separate lables plus one for coordinate input. This makes one longer program, but is actually shorter than the total of the other five needed the other way.

The illustration to the right allows using all four of the intersection types for trying out the programming after you have input it. We have left everything in the 'bearing' format, rather than refer to 'directions' and distances, but you may work in either azimuth or bearing for your input and output.

Input for the program begins with prompts for the beginning and ending coordinate pairs, in this case N_1, E_1 and N_2, E_2. The point to be output as the intersection will be N_3, E_3 for the bearing-bearing, bearing-distance and the distance-distance solutions.



For all of the distance/offset solutions, the intersection point will be somewhere along the bearing line from points 1 to 3, or that same line produced past 3, shown as N_6,E_6, N_7,E_7 and N_8, E_8.

The step-by-step program input/output instructions begin on the next page. We've used the program keystroke instructions to solve some of the various problems as shown in the table to the right. The user should solve the remaining Bearing-Offset problems for practice.

This is a program where we suggest that you input part of it, take a break, then complete the rest at a different sitting, to help reduce program input errors.

1	150.00	175.00
2	125.00	400.00
3	To be calculated	
4	300.00	200.00
5	400.00	150.00
6	To be calculated	
7	To be calculated	
8	To be calculated	

PROGRAM: Intersections

	PROMPT	INSTRUCTIONS	KEYSTROKES	OUTPUT
1		Begin the intersection program	[XEQ] [I]	
3	BEG NORTHING:	Input the North coordinate of the beginning point.	[R/S]	
3	BEG EASTING:	Input the East coordinate of the beginning point.	[R/S]	
3	END NORTHING:	Input the North coordinate of the ending point.	[R/S]	
4	END EASTING:	Input the East coordinate of the ending point.	[R/S]	
5	AZ=0 BRG=1	**Selection Prompt** To work in Azimuth mode, input 0. To work in Bearing/Quad mode input 1.	[0] or [1] then [R/S]	
6	AZ-AZ=1 OR BRG-BRG=1 Depending on 5	**If Azimuth mode was selected** OR **If Bearing/Quad mode was selected**	To select this type, input [1], to NOT select this type, NO input. Stroke [R/S]	
7	AZ-DIST=1 OR BRG-DIST=1 Depending on 5	**If Azimuth mode was selected** OR **If Bearing/Quad mode was selected**	To select this type, input [1], to NOT select this type, NO input. Stroke [R/S]	
8	DIST-DIST=1	**To select this type, input** 1 **to NOT select this type, NO input.**	[R/S]	
9	AZ-OS=1 OR BRG-OS=1 Depending on 5	**If Azimuth mode was selected** OR **If Bearing/Quad mode was selected**	To select this type, input [1], to NOT select this type, NO input. Stroke [R/S]	

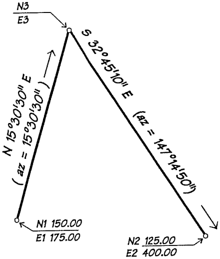
At this point you have input the beginning and ending coordinates, selected the type of input (azimuth or bearing) and the type of intersection you want to do.

Separate instructions will be given for each type of intersection solution (on the following pages), as examples, and will use either azimuth or bearing input. If you chose a different input than that shown in the example, the only difference will be the prompts for the directions.

Output, in all cases will be the bearing (or azimuth) and distance from the beginning point to the intersection point, the coordinates of the intersection , then bearing (or azimuth) and distance to the ending point.

PROGRAM: Intersections (continued) BEARING – BEARING Selected

Working the problem as an input example, use the figure below for the input.

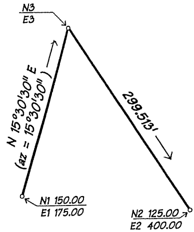


This example uses the input as bearing and quad code with the matching output.

	PROMPT	KEYSTROKES	OUTPUT
1a	BEARING:	[1][5][.][3][0][3] [R/S]	
1b	QUAD CODE:	[1] [R/S]	
2a	BEARING:	[3][2][.][4][5][1][R/S]	
2b	QUAD CODE:	[2][R/S]	B= 15.3030
3		[R/S]	Q= 1.0000
4		[R/S]	D = 235.4673
5		[R/S]	N= 376.8943
		[R/S]	E= 237.9589
		[R/S]	D= 299.5130
		[R/S]	B= 32.4510
		[R/S]	Q= 2.0000
		[R/S] Returns to the intersection program for new calculations. To leave the program stroke [C]	

PROGRAM: Intersections (continued) BEARING – DISTANCE Selected

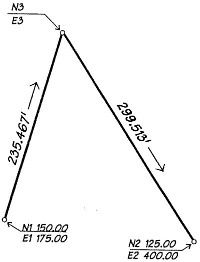
Again working the problem as an input example, use Azimuth for the input on the figure below.



	PROMPT	KEYSTROKES	OUTPUT
1	AZIMUTH:	[1][5][.][3][0][3] [R/S]	
1b	DISTANCE:	[2][9][9][.][5][1][3][R/S]	A= 15,3030
2			D= 235.4672
3		[R/S]	N= 376.8943
4		[R/S]	E= 237.9589
5		[R/S]	A= 147.1450
6		[R/S]	D= 299.5130
		[R/S] Returns to the intersection program for new calculations. To leave the program stroke [C]	

PROGRAM: Intersections (continued) DISTANCE – DISTANCE Selected

Working the problem as an input example, use the figure below and select azimuth for the directions.

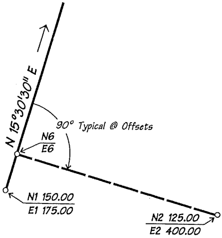


	PROMPT	KEYSTROKES	OUTPUT
1	DISTANCE:	[2][3][5][.][4][6][7][R/S]	
2	DISTANCE:	[2][9][9][.][5][1][3][R/S]	A= 15.3030
3		[R/S]	D= 235.4670
3		[R/S]	N= 376.8943
4		[R/S]	E= 237.9589
5		[R/S]	A= 147.1450
6		[R/S]	D= 299.5130
		[R/S] Returns to the intersection program for new calculations. To leave the program stroke [C]	

NOTE: The last two types may have 2 possible solutions. For the 2nd solution in Bearing-Distance, just reverse the direction of the bearing and re-run the calculation. For the 2nd solution on the Distance-Distance type, run it backwards.

PROGRAM: Intersections (continued) BEARING – OFFSET Selected

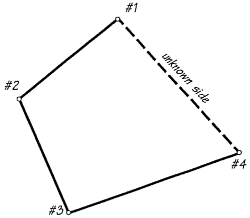
For this one, we've calculated the offset to the second point as the example (N6, E6). We're going to use the other two in the exercise.



	PROMPT	KEYSTROKES	OUTPUT
1a	AZIMUTH:	[1][5][.][3][0][3] [R/S]	A= 15.3030
1b		[R/S]	D= 36.0704
2		[R/S]	N= 184.7571
3		[R/S]	E= 184.6444
4a		[R/S]	A= 105.3030
4b		[R/S]	D= 223.4926
		[R/S] Returns to the intersection program for new calculations. To leave the program stroke [C]	

calculating missing sides

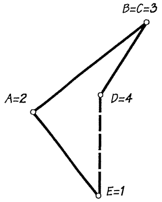
It sometimes happens that some of the dimensions for a traverse (or deed) are missing. In order to solve for *any* missing part of a traverse, we have to first **assume** that the traverse closes, because any part we calculate is based upon the information furnished by the other parts, and will only work on a *closed* figure. When we calculate our answer, we are *forcing* a closure.



One side missing is the most obvious example of solving for a missing part. We have everything except the closing line.

If, using the traverse program, coordinates are put on each of the known corners, it is a simple matter to obtain the missing side's length and direction by inversing from #4 to #1.

Adjacent missing Parts. It also happens that you are missing TWO parts of a traverse and need to simultaneously solve for both parts. You can do this by reducing the traverse to a point where the difference in latitudes and departures of the *known* parts may be used to solve the two missing sides.



The illustration to the left shows the basic principles involved, and we can use the formulas

$$a = \frac{y(\sin B) - x(\cos B)}{\sin(B - A)}$$

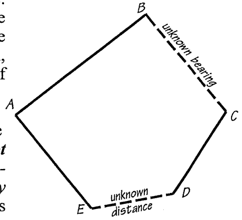
or

$$b = \frac{x(\cos A) - y(\sin A)}{\sin(B - A)}$$

These formulas solve for missing lengths of two adjacent sides, and may also be re-written in the form

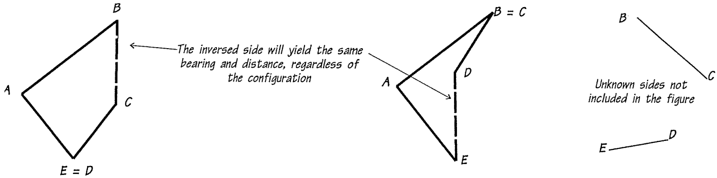
$$\sin(B - A) = \frac{y(\sin B) - x(\cos B)}{a} = \frac{x(\cos A) - y(\sin A)}{b}$$

when we have the sides and need to solve for the missing bearings. If you look at these formulas, you will notice that they are the formulas for doing a bearing-bearing and bearing-distance intersection. Additional needed information, from that point on, may be solved through the use of the Law of Sines. The Law of Cosines is used for solving distance-distance intersections.

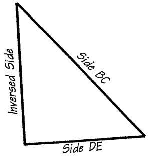


Non-adjacent unknowns also occur. It may be that the missing parts will fall on sides of the traverse which are ***not*** adjacent to each other. You can arrive at a solution by *re-arrangement* of the traverse. For purposes of *temporary* coordinate values you can connect the sides with known bearings AND distances together, making the missing sides adjacent.

The first step is to ignore the unknown courses and connect up all of the courses with *both* known bearings and distances. Simply rearrange the figure, leaving out the unknown sides.



Calculate coordinates for the points in the rearranged figure and inverse for a closure (above). Next, the inversed side, combined with the two sides that contain unknowns, will form a figure like the one shown to the right. *This* figure may now be solved as a triangle, or as an intersection problem (in the case illustrated, use bearing-distance).



Once the unknowns are resolved, re-assemble the original figure and calculate actual coordinates for the angle points.

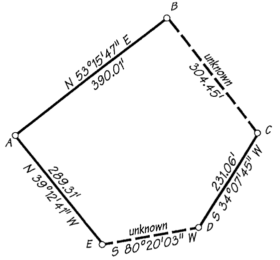
So, how would something like this happen in the first place? Consider the following exerpt from a deed written about 50 years ago, when all of the properties in the area were farm lands:

> . . . Thence, N 22°25' E 342.67 feet to a point in that line common to the property now or formerly owned by George W. Brown, as shown in that certain Grant Deed recorded May 16, 1923 in Book 243 of Official Records of Bohunk County at page 22; thence along said common line 435.96 feet to the most westerly line of the Smith property as shown in that certain Quitclaim Deed recorded September 24, 1940 in Book 136 of Official Records of Bohunk County at page 209; thence, along last said westerly line S 47°22' W to the point of beginning, containing . . .

And there you have a not uncommon source of unknown lines. The distance along Brown's line is known but not the direction, and Smith's line has a direction but not a distance.

When solving a problem like this it's important to understand that the answers (as in the simple inverse for closure) are a 'forced' solution. It is only as accurate as the other information that was used to calculate it. The answer is not 'real' by any definition but is a solution based on the known information. A look at Brown's deed might give a bearing for that line, but was written at a different time and not necessarily based on the same basis of bearings. Without more information being known there is no way to check the answers.

Exercise 8

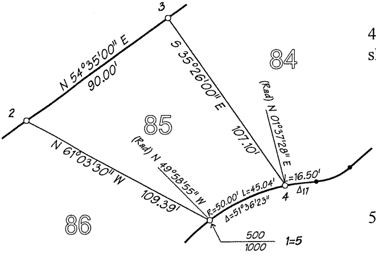


Using the figure shown to the right, solve for the unknowns, then calculate and close the boundary.

1. What is the bearing of line B-C? ____________

2. What is the distance for lint D-E? ____________

3. What is the area of the enclosed property? ____________



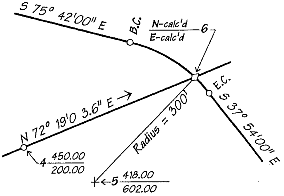
4. Calculate the coordinates for Lot 85, shown to the left.

Pt #	North Coordinate	East Coordinate
2		
3		
4		

5. What is the area of Lot 85?

____________ Sq. Ft.

6. Calculate the coordinates for point #6, the B.C and E.C. of the curve in the figure to the right..



7. Calculate the curve data for the curve

Radius = ____________

Delta = ____________

Length = ____________

Tangent = ____________

Chord = ____________

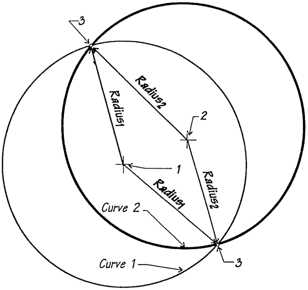
External = ____________

Mid-Ordinate = ____________

8. If the station at the B.C. is 115+24.96, what is the station at the intersection of the curve and line?

____

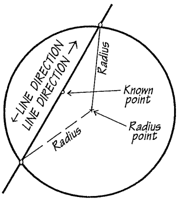
curve to curve intersection



The intersection program will calculate the intersection point along the arcs of two curves when the two radii and the point numbers of the radius points are known. Use the Distance-Distance option, with the radii being the two distances, to solve the problem.

Input is ALWAYS counter-clockwise, as seen from the intersection point.

line to curve intersection



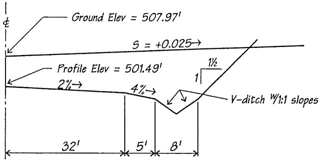
You need to know the radius of the curve, the coordinates of its radius point, a known coordinate at any location on the line, and the direction of the line.

If the line point is on the *inside* of the curve, the direction of the line is given **toward** the point of intersection that you want to calculate.. If *outside* the curve, it is often easiest to set a point on line that is inside the curve to avoid confusion.

Both of these problems can also be solved by calculation as triangle problems, although the intersection is quicker.

slope staking

Surveyors tend to think of slope staking as a trial-and-error field exercise, as it is done in the field. On a test it has to be generalized, and unless the illustration includes a grid at a given scale so you can count the squares for distance and elevation, the quick way is to use the intersection program. They will have to give you a picture with all of the slopes, distances, etc. to even formulate a problem.



Okay, turn the slopes into azimuths, calculate the distance to the far side of the v-ditch, work the slopes to get an elevation for that point.

Using the elevations as the north coordinates and the distances as the east coordinates, calculate the intersection at the catch point. It's just an azimuth-azimuth intersection problem.

trouble-shooting your programs

If you are running a program and encounter an error message, stop and stroke [↰] [R/S] to go into program mode. The calculator will have stopped at the point that is causing the problem.

```
RPN
NONEXISTENT
```

In the case shown, "NONEXISTENT" will indicate that you have a label missing . . . You didn't input a program that the calculator is looking for yet. "DIVIDE BY 0" would usually mean that an equation or program step is trying to divide by an empty register, and "INVALID EQN" tells you that something is wrong with an equation, maybe an extra parentheses or a missing one. Correct the error and try running the program again. If the program is correct, the LN and CK numbers (see chart on page 44) should be the same as shown in the chart.

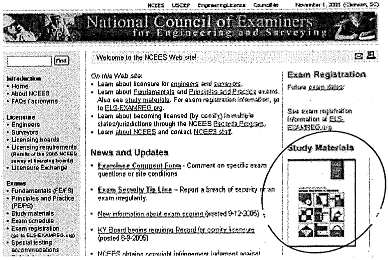
If you need to take a break while programming, stroke [C] to leave program mode and then stroke [↰] [C] to turn the calculator off. When you come back, turn the calculator back on and stroke [↰] [R/S] to go into program mode. You will be at the same place you left off.

During a programming session you can go to any particular step by stroking [↰] [XEQ] [.]. This brings up a prompt display like the one to the right. Type in the step you want to go to (i.e. S0127 and when you go into program mode you will be at that step.

```
RPN
GTO _____
```

NCEES

The NCEES examinations are closed-book, but reference material (formulas and information) are provided in both the morning and afternoon examination booklets. You can download the same reference materials for study from www.ncees.org before the test . This information is free, and is in PDF format that you can download to your computer or print out.



Click on Study Materials to select the fundamental exams. A free download is available for both the Surveying and Engineering exam materials

You'll also find that a book of sample test questions is available at a low cost. Once you have the information (it's usually updated prior to each test) you'll note that a lot of the material is already included in the programs in this book. For what isn't included you have some options. **Read**

chapters 6 and 7 in your user's manual and you'll find that any of the equations that are listed, photogrammetry for instance, can easily be input into the calculator's equation library, and ready for solution by the solver, or just by executing the equation. Go into equation mode by stroking [↰] [STO], and you will be at the top of the equation list.

Typing in the given equations as you need them during the test only takes a minute or two, the calculator will prompt you for each value and solve it for you, avoiding errors that you might make in working with the numbers directly.

As a general rule, the test will vary with each testing session, but in each test you will be given the materials you need to answer the problems. It would be a good use of your time in preparing to take the test to be as familiar with your calculator as you can. Learn how to use the statistics functions, as well as the equation functions.

The calculator's manual even includes a program for Time Value of Money (Chapter 17, Page 1) and probability functions (Chapter 4, Page 14) to handle some of the test questions you may be least familiar with. Again, the more familiar you are with your calculator, the better you'll do in the test.

answer key

Exercise 1 (page 6)
1. 147°24'23" 2. 90°53'15" 3. N 40°07'18" E (az = 40°07'18") 4. S 82°58'18" W (az = 262°58'18")

Exercise 2 (page 6)
1. 117°41'34" 2. 140°18'31" 3. 63°54'38" 4. 117°35'20" 5. 80°13'11" 6. S 04°20'46" E
7. S 78°09'18" E 8. 223°49'17" 9. .9550 10. -3.7652 11. .9994 12. -17.1423

Exercise 3 (page 10)

	1.		2.		3.		4.
Radius =	**510.23**	Radius =	**400.00**	Radius =	**200.00'**	Radius =	16,127.45
Delta =	27°45'25"	Delta =	34°44'59"	Delta =	2°56'21"	Delta =	**1°25'16"**
Length =	247.18'	Length =	242.60'	Length =	**10.26'**	Length =	400.01'
Tangent =	126.07'	Tangent =	**125.16'**	Tangent =	5.13'	Tangent =	200.02'
Chord =	**244.77'**	Chord =	238.90'	Chord =	10.26'	Chord =	**400.00'**
External =	15.34'	External =	19.12'	External =	0.07'	External =	1.24'
Mid-Ordinate =	14.90'	Mid-Ordinate =	18.25'	Mid-Ordinate =	0.07'	Mid-Ordinate =	1.24,
		Sector =	48,519.88□'	Sector =	1,026.00□'		
		Segment =	2,920.36□'	Segment =	0.45□'		
		Fillet =	1,544.12□'	Fillet =	0.23□'		

Exercise 4 (page 13)

1.

		STATION	DEFLECTION	CHORD
Radius =	510.23'	12+19.23 B.C.	0°	0.00'
Delta =	27°45'25"	12+50	1°43'40"	30.77'
Length =	247.18	13+00	4°32'06"	80.69'
Tangent =	126.06	13+50	7°20'32"	130.41'
Chord =	244.77'	14+00	10°08'59"	179.83'
External =	15.34	14+50	12°57'25"	228.81'
Mid-Ordinate =	14.89	14+66.4 EC.	13°52'42"	244.77'

2.

		STATION	TANGENT DIST.	OFFSET
Radius =	**400.00'**	**122+34.97**	0.00'	0.00'
Delta =	34°44'59"	**122+50**	15.03'	0.28'
Length =	242.60'	**123+00**	64.74'	5.27'
Tangent =	**125.16'**	**123+50**	113.45'	16.43'
Chord =	238.90'	**124+00**	160.39'	33.56'
External =	19.12'	**124+50**	204.82'	56.42'
Mid-Ordinate =	18.25'	124+77.57 **E.C.**	228.00'	71.34'

3.

		STATION	CHORD DIST.	OFFSET
Radius =	386.31'	**53+24.37**	0.00'	0.00'
Delta =	**35°15'22"**	**53+50**	24.67'	6.95'
Length =	**237.71'**	**54+00**	73.85'	15.72'
Tangent =	**122.75'**	**54+50**	123.76'	18.08'
Chord =	233.98'	**55+00**	173.56'	13.98'
External =	19.03	**55+50**	222.41'	3.48'
Mid-Ordinate =	18.14	55+62.08 **E.C.**	233.98'	0.00'

Exercise 5 (pages 19 and 20)

1.

Side 1	**12.163**
Angle 1	**76°03'56"**
Side 2	**18.364**
Angle 2	37°24'35"
Side 3	19.432
Angle 3	66°31'29"
Area	108.3941

2.

Side 1	**29.682**
Angle 1	75°14'51"
Side 2	**23.9162**
Angle 2	**60°19'18"**
Side 3	33.0374
Angle 3	44°25'51"
Area	343.240

3.

Side 1	**15**
Angle 1	**84°28'18"**
Side 2	21.8384
Angle 2	**36°12'30"**
Side 3	25.2745
Angle 3	59°19'12"
Area	163.0263

4.

Side 1	**65.467**
Angle 1	**84°28'18"**
Side 2	109.4502
Angle 2	32°17'00"
Side 3	122.0028
Angle 3	**63°14'42"**
Area	3,566.0253

5.

Side 1	**18.249**
Angle 1	**79°22'06"**
Side 2	24.6097
Angle 2	**40°10'30"**
Side 3	27.8020
Angle 3	60°27'24"
Area	220.6965

6.

Side 1	**32.416**
Angle 1	**87°13'51"**
Side 2	38.1001
Angle 2	41°32'56"
Side 3	48.8168
Angle 3	**51°13'13"**
Area	616.8048

Exercise 6 (page 25)

1. BVC @ Station 12+05.56, Elevation BVC 124.11,
   high point Station 14+27.78, elevation 126.33, EVC @ Station 16+05.56, Elevation 126.33
2. 12+00 _124.00__ 12+50 __124.91__ 13+00 __125.60___ 13+50 __126.06__ 14+00 __124.91___

14+50 __126.31__ 15+00 ___126.10__ 15+50 _125.66___ 16+00 _125.00___

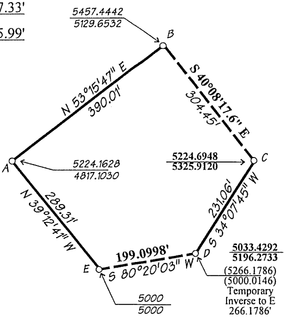
3. elevation 123.58 @ __11+79.00__ and 121.56 @ __10+78.00__ (neither one is in the vertical curve)

Exercise 7 (page 32)

To the southwest Lot Corner azimuth = 76°14'26", distance = 117.43'

To the southeast Lot Corner azimuth = 88°42'15", distance = 207.33'

To the building corner azimuth = 68°32'26", distance = 145.99'

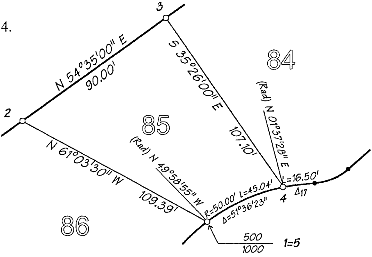


Exercise 8 (page 39)

Assume that the coordinates for point E are 5000, 5000

1. What is the bearing of line B-C? **S 40°08'17.6" E**
2. What is the distance for line D-E? **199.0998'**
3. What is the area of the enclosed property? **132,944.6772 sq ft**

4.

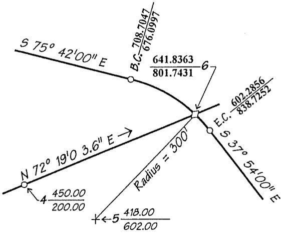


Pt #	North Coordinate	East Coordinate
2	552.9359	904.2714
3	605.0925	977.6177
4	517.8285	1039.7095

5. What is the area of Lot 85?

   ____6577.70____ Sq. Ft.

Exercise 8 *(Continued)*



6. Calculate the coordinates for point #6, the B.C and E.C. of the curve in the figure to the right.

7. Calculate the curve data for the curve

Radius =	300.00
Delta =	37°48'00"
Length =	197.92'
Tangent =	102.71'
Chord =	194.35
External =	17.10'
Mid-Ordinate =	16.17'

8. If the station at the B.C. is 115+24.96, what is the station at the intersection of the curve and line?

116+68.66

LBL U . . . for "undoing flag settings"

This one was suggested by a user, and is added in, here at the end of the book, because it is a great idea. There are times when one or more flags are left set after using a program and can affect the next program you want to use.

To insure that you have all flags clear before starting a new calculation, simply stroke [XEQ] [U]. The program quickly clears all used flags that might still be set. LN=24 CK=2701

U0001 LBL U	[left-shift] [+] [U]
U0002 CLx	[left-shift] [←] [1]
U0003 CF 0	[right-shift] [×] [2] [0]
U0004 CF 1	[right-shift] [×] [2] [1]
U0005 CF 2	[right-shift] [×] [2] [2]
U0006 CF 3	[right-shift] [×] [2] [3]
U0007 CF 4	[right-shift] [×] [2] [4]
U0008 RTN	[right-shift] [+]

We would appreciate your comments and suggestions regarding the workbook and programs. If you used the hp33s calculator during a test, we would like to know whether or not you feel it saved time or otherwise was helpful. Please DO NOT send us test examples from your test.

Software by D'Zign
P.O. Box 430
Tollhouse, CA 93667

Lbl	Program Name	Pgm page	LN	Ck Sum	Registers Used																									
					A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	**Azimuth to Bearing**	4	17	B086														■			■									
B	**Bearing to Azimuth**	5	108	B010		■															■									
C	**Circular Curves**	P1	891	B70A			■	■					■			■	■					■	■	■						
D	**Add/Subtract D.ms**	6	27	0AA4																										
E	**Elevation Known**	P9	174	E894	■				■			■	■		■	■			■			■	■	■					■	
F	*not used*																													
G	**Triangle Solutions**	P4	1198	6FF6	■			■	■	■	■	■	■							■				■						
H	sub to Elev. Known	P10	69	E383					■			■	■									■	■							
I	**Intersections**	P13	856	EC20	■		■	■			■	■		■	■	■	■	■	■	■	■		■	■	■			■	■	■
J	sub to Traverse	P11	122	DC30	■	■		■	■		■					■	■	■	■	■	■		■		■			■	■	■
K	**Close Traverse**	P11	179	F0C1	■	■		■	■		■	■						■		■	■		■					■	■	■
L	**Inverse**	P12	93	8F6E	■	■		■				■	■	■	■						■								■	■
M	sub to intersections	P15	36	7301	■	■															■									
N	entry point Az-Brg	4	81	1DE5		■															■									
O	**Curve Layout**	P3	169	EF34	■	■	■	■		■	■	■	■		■	■	■		■			■	■	■						
P	**Vertical Intersection**	P9	221	0172					■	■			■				■	■	■				■	■				■		
Q	sub to Curve Layout	P3	310	7BB3	■	■	■	■		■	■	■	■		■	■	■		■			■	■	■						
R	*sub to inverse*	P12	151	15AA																										
S	sub to triangle	P7	309	ECF2	■			■	■	■	■	■	■							■				■						
T	**Traverse**	P11	99	63A8	■	■		■	■		■					■	■	■	■	■	■		■		■			■	■	■
U	*For clearing flags*	45	24	2701																										
V	**Vertical Curve**	P8	142	E22F					■				■			■			■			■	■							
W	sub to intersections	P15	32	21B6	■	■		■	■		■					■	■	■	■	■	■		■		■			■	■	■
X	sub to vertical curve	P8	94	42F2					■				■			■			■			■	■							
Y	sub to vertical curve	P8	61	7C16					■				■			■			■			■	■							
Z	sub to vertical curve	P8	66	9CB9					■				■			■	■		■			■	■							
					A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z

– Circular Curves

C0001 LBL C	[↰] [+] [C]
C0002 CF 1	[↱] [×] [2] [1]
C0003 CF 2	[↱] [×] [2] [2]
C0004 CF 3	[↱] [×] [2] [3]
C0005 CLVARS	[↰] [←] [2]
C0006 SF 10	[↱] [×] [1] [.] [0]
C0007 DELTA:	[↱] [STO] *then* [RCL] *before each alpha input*
C0008 CF 10	[↱] [×] [2] [.] [0]
C0009 →HR	[↰] [5]
C0010 STO D	[STO] [D]
C0011 X≠0?	[↱] [÷] [1]
C0012 SF 1	[↱] [×] [1] [1]
C0013 INPUT R	[↰] [0] [R]
C0014 X≠0?	[↱] [÷] [1]
C0015 SF 2	[↱] [×] [1] [2]
C0016 RCL D	[RCL] [D]
C0017 RCL R	[RCL] [R]
C0018 x	[×]
C0019 X≠0?	[↱] [÷] [1]
C0020 CF 1	[↱] [×] [2] [1]
C0021 X≠0?	[↱] [÷] [1]
C0022 CF 2	[↱] [×] [2] [2]
C0023 X≠0?	[↱] [÷] [1]
C0024 SF 3	[↱] [×] [1] [3]
C0025 INPUT L	[↰] [0] [L]
C0026 FS? 2	[↱] [×] [3] [2]
C0027 D=Lx180÷(πxR)	[↱] [STO] *then* [RCL] *before each alpha input*
C0028 FS? 2	[↱] [×] [3] [2]
C0029 ABS	[↰] [y^x]
C0030 FS? 2	[↱] [×] [3] [2]
C0031 STO D	[STO] [D]
C0032 X≠0?	[↱] [÷] [1]
C0033 CF 2	[↱] [×] [2] [2]
C0034 FS? 1	[↱] [×] [3] [1]
C0035 R=Lx180÷(πxD)	[↱] [STO] *then* [RCL] *before each alpha input*
C0036 FS? 1	[↱] [×] [3] [1]
C0037 ABS	[↰] [y^x]
C0038 FS? 1	[↱] [×] [3] [1]
C0039 STO R	[STO] [R]
C0040 X≠0?	[↱] [÷] [1]
C0041 CF 1	[↱] [×] [2] [1]
C0042 INPUT C	[↰] [0] [C]

Continued next column

C0043 FS? 2	[↱] [×] [3] [2]
C0044 2xASIN(C÷(Rx2))	[↱] [STO] *then* [RCL] *before each alpha input*
C0045 FS? 2	[↱] [×] [3] [2]
C0046 ABS	[↰] [y^x]
C0047 FS? 2	[↱] [×] [3] [2]
C0048 STO D	[STO] [D]
C0049 X≠0?	[↱] [÷] [1]
C0050 CF 2	[↱] [×] [2] [2]
C0051 FS? 1	[↱] [×] [3] [1]
C0052 C÷(2xSIN(D÷2))	[↱] [STO] *then* [RCL] *before each alpha input*
C0053 FS? 1	[↱] [×] [3] [1]
C0054 ABS	[↰] [y^x]
C0055 FS? 1	[↱] [×] [3] [1]
C0056 STO R	[STO] [R]
C0057 X≠0?	[↱] [÷] [1]
C0058 CF 1	[↱] [×] [2] [1]
C0059 INPUT T	[↰] [0] [T]
C0060 FS? 2	[↱] [×] [3] [2]
C0061 D=ATAN(T÷R)x2	[↱] [STO] *then* [RCL] *before each alpha input*
C0062 FS? 2	[↱] [×] [3] [2]
C0063 ABS	[↰] [y^x]
C0064 FS? 2	[↱] [×] [3] [2]
C0065 STO D	[STO] [D]
C0066 X≠0?	[↱] [÷] [1]
C0067 CF 2	[↱] [×] [2] [2]
C0068 FS? 1	[↱] [×] [3] [1]
C0069 R=T÷TAN(0.5xD)	[↱] [STO] *then* [RCL] *before each alpha input*
C0070 FS? 1	[↱] [×] [3] [1]
C0071 ABS	[↰] [y^x]
C0072 FS? 1	[↱] [×] [3] [1]
C0073 STO R	[STO] [R]
C0074 X≠0?	[↱] [÷] [1]
C0075 CF 1	[↱] [×] [2] [1]
C0076 CF 1	[↱] [×] [2] [1]
C0077 CF 2	[↱] [×] [2] [2]
C0078 RCL L	[RCL] [L]
C0079 X≠0?	[↱] [÷] [1]
C0080 SF 1	[↱] [×] [1] [1]
C0081 RCL C	[RCL] [C]
C0082 X≠0?	[↱] [÷] [1]
C0083 SF 2	[↱] [×] [1] [2]

Continued on next page

Continued from previous page

C0084	RCL T	[RCL] [T]
C0085	X≠0?	[↱] [÷] [1]
C0086	SF 3	[↱] [×] [1] [3]
C0087	FS? 1	[↱] [×] [3] [1]
C0088	T=RxTAN(D÷2)	[↱] [STO] *then* [RCL] *before each alpha input*
C0089	FS? 1	[↱] [×] [3] [1]
C0090	ABS	[↰] [y^x]
C0091	FS? 1	[↱] [×] [3] [1]
C0092	STO T	[STO] [T]
C0093	FS? 1	[↱] [×] [3] [1]
C0094	C=2xRx(SIN(D÷2))	[↱] [STO] *then* [RCL] *before each alpha input*
C0095	FS? 1	[↱] [×] [3] [1]
C0096	ABS	[↰] [y^x]
C0097	FS? 1	[↱] [×] [3] [1]
C0098	STO C	[STO] [C]
C0099	L=RxDx(π÷180)	[↱] [STO] *then* [RCL] *before each alpha input*
C0100	FS? 2	[↱] [×] [3] [2]
C0101	ABS	[↰] [y^x]
C0102	FS? 2	[↱] [×] [3] [2]
C0103	STO L	[STO] [L]
C0104	FS? 2	[↱] [×] [3] [2]
C0105	T=RxTAN(D÷2)	[↱] [STO] *then* [RCL] *before each alpha input*
C0106	FS? 2	[↱] [×] [3] [2]
C0107	ABS	[↰] [y^x]
C0108	FS? 2	[↱] [×] [3] [2]
C0109	STO T	[STO] [T]
C0110	FS? 3	[↱] [×] [3] [3]
C0111	L=RxDx(π÷180)	[↱] [STO] *then* [RCL] *before each alpha input*
C0112	FS? 3	[↱] [×] [3] [3]
C0113	ABS	[↰] [y^x]
C0114	FS? 3	[↱] [×] [3] [3]
C0115	STO L	[STO] [L]
C0116	FS? 3	[↱] [×] [3] [3]
C0117	C=2xRxSIN(D÷2)	[↱] [STO] *then* [RCL] *before each alpha input*
C0118	FS? 3	[↱] [×] [3] [3]
C0119	ABS	[↰] [y^x]
C0120	FS? 3	[↱] [×] [3] [3]
C0121	STO C	[STO] [C]
C0122	FS? 3	[↱] [×] [3] [3]

Continued next column

C0123	RCL R	[RCL] [R]
C0124	FS? 3	[↱] [×] [3] [3]
C0125	TAN(D÷2)	[↱] [STO] *then* [RCL] *before each alpha input*
C0126	FS? 3	[↱] [×] [3] [3]
C0127	x	[×]
C0128	FS? 3	[↱] [×] [3] [3]
C0129	ABS	[↰] [y^x]
C0130	FS? 3	[↱] [×] [3] [3]
C0131	STO T	[STO] [T]
C0132	RCL D	[RCL] [D]
C0133	→HMS	[↱] [5]
C0134	STO I	[STO] [I]
C0135	VIEW I	[↱] [0] [I]
C0136	VIEW R	[↱] [0] [R]
C0137	VIEW L	[↱] [0] [L]
C0138	VIEW C	[↱] [0] [C]
C0139	VIEW T	[↱] [0] [T]
C0140	E=TxTAN(D÷4)	[↱] [STO] *then* [RCL] *before each alpha input*
C0141	ABS	[↰] [y^x]
C0142	STO E	[STO] [E]
C0143	VIEW E	[↱] [0] [E]
C0144	M=Rx(1-COS(D÷2))	[↱] [STO] *then* [RCL] *before each alpha input*
C0145	ABS	[↰] [y^x]
C0146	STO M	[STO] [M]
C0147	VIEW M	[↱] [0] [M]
C0148	SF 10	[↱] [×] [1] [.] [0]
C0149	AREAS:	[↱] [STO] *then* [RCL] *before each alpha input*
C0150	PSE	[↱] [R/S]
C0151	π	[↱] [COS]
C0152	RCL R	[RCL] [R]
C0153	x^2	[x^2]
C0154	x	[×]
C0155	RCL D	[RCL] [D]
C0156	360	[3] [6] [0]
C0157	÷	[÷]
C0158	x	[×]
C0159	STO S	[STO] [S]
C0160	0	[0]
C0161	x<>y	[x↔y]
C0162	SECTOR:	[↱] [STO] *then* [RCL] *before each alpha input*

Continued on next page

Continued from previous page

C0163 PSE	[↱] [R/S]
C0164 STOP	[R/S]
C0165 RCL D	[RCL] [D]
C0166 2	[2]
C0167 ÷	[÷]
C0168 COS	[COS]
C0169 RCLx R	[RCL] [×] [R]
C0170 RCLx C	[RCL] [×] [C]
C0171 2	[2]
C0172 ÷	[÷]
C0173 -	[−]
C0174 0	[0]
C0175 x<>y	[x↔y]
C0176 SEGMENT:	[↱] [STO] *then* [RCL] *before each alpha input*
C0177 PSE	[↱] [R/S]
C0178 STOP	[R/S]
C0179 RCL R	[RCL] [R]
C0180 RCLx T	[RCL] [×] [T]
C0181 RCL- S	[RCL] [−] [S]
C0182 0	[0]
C0183 x<>y	[x↔y]
C0184 FILLET:	[↱] [STO] *then* [RCL] *before each alpha input*
C0185 PSE	[↱] [R/S]
C0186 STOP	[R/S]
C0187 MORE=0 STAKE=1	[↱] [STO] *then* [RCL] *before each alpha input*
C0188 X=0?	[↱] [÷] [6]
C0189 GTO C	[↰] [XEQ] [C]
C0190 X≠0?	[↱] [÷] [1]
C0191 GTO O	[↰] [XEQ] [O]

– Curve Layout

O0001 LBL O	[↰] [+] [O]
O0002 CF 0	[↱] [×] [2] [0]
O0003 CF 1	[↱] [×] [2] [1]
O0004 CF 2	[↱] [×] [2] [2]
O0005 CF 3	[↱] [×] [2] [3]
O0006 SF 10	[↱] [×] [1] [.] [0]
O0007 SELECT TYPE	[↱] [STO] *then* [RCL] *before each alpha input*
O0008 PSE	[↱] [R/S]
O0009 CLx	[↰] [←] [1]
O0010 DEFLECTION	[↱] [STO] *then* [RCL] *before each alpha input*

Continued next column

O0011 x≠0?	[↱] [÷] [1]
O0012 SF 1	[↱] [×] [1] [1]
O0013 CLx	[↰] [←] [1]
O0014 TAN-OS	[↱] [STO] *then* [RCL] *before each alpha input*
O0015 x≠0?	[↱] [÷] [1]
O0016 SF 2	[↱] [×] [1] [2]
O0017 CLx	[↰] [←] [1]
O0018 CHD-OS	[↱] [STO] *then* [RCL] *before each alpha input*
O0019 x≠0?	[↱] [÷] [1]
O0020 SF 3	[↱] [×] [1] [3]
O0021 CLx	[↰] [←] [1]
O0022 OFFSET=:	[↱] [STO] *then* [RCL] *before each alpha input*
O0023 STO O	[STO] [O]
O0024 RCL+ R	[RCL] [+] [R]
O0025 RCL÷ R	[RCL] [÷] [R]
O0026 STO K	[STO] [K]
O0027 BC STATION:	[↱] [STO] *then* [RCL] *before each alpha input*
O0028 STO B	[STO] [B]
O0029 RCL+ L	[RCL] [+] [L]
O0030 STO H	[STO] [H]
O0031 RCL D	[RCL] [D]
O0032 2	[2]
O0033 ÷	[÷]
O0034 RCL÷ L	[RCL] [÷] [L]
O0035 STO F	[STO] [F]
Q0001 LBL Q	[↰] [+] [Q]
Q0002 INPUT STATION:	[↱] [STO] *then* [RCL] *before each alpha input*
Q0003 RCL- B	[RCL] [−] [B]
Q0004 STO J	[STO] [J]
Q0005 RCLx F	[RCL] [×] [F]
Q0006 STO A	[STO] [A]
Q0007 SIN	[SIN]
Q0008 RCLx R	[RCL] [×] [R]
Q0009 2	[2]
Q0010 x	[×]
Q0011 STO C	[STO] [C]
Q0012 FS? 1	[↱] [×] [3] [1]
Q0013 DEF ANGLE	[↱] [STO] *then* [RCL] *before each alpha input*
Q0014 FS? 2	[↱] [×] [3] [2]
Q0015 TAN DIST	[↱] [STO] *then* [RCL] *before each alpha input*

Continued on next page

Continued from previous page

Q0016	FS? 3	[↱][×][3][3]
Q0017	CHD DIST	[↱][STO] *then* [RCL] *before each alpha input*
Q0018	CF 10	[↱][×][2][·][0]
Q0019	FS? 1	[↱][×][3][1]
Q0020	→HMS(A)	[↱][STO] *then* [RCL] *before each alpha input*
Q0021	FS? 2	[↱][×][3][2]
Q0022	Kx(CxCOS(A))	[↱][STO] *then* [RCL] *before each alpha input*
Q0023	FS? 3	[↱][×][3][3]
Q0024	KxCxCOS((D÷2)-A)	[↱][STO] *then* [RCL] *before each alpha input*
Q0025	0	[0]
Q0026	x<>y	[x↔y]
Q0027	STOP	[R/S]
Q0028	SF 10	[↱][×][1][·][0]
Q0029	FS? 1	[↱][×][3][1]
Q0030	CHORD	[↱][STO] *then* [RCL] *before each alpha input*
Q0031	FS? 2	[↱][×][3][2]
Q0032	OFFSET	[↱][STO] *then* [RCL] *before each alpha input*
Q0033	FS? 3	[↱][×][3][3]
Q0034	OFFSET	[↱][STO] *then* [RCL] *before each alpha input*
Q0035	CF 10	[↱][×][2][·][0]
Q0036	FS? 1	[↱][×][3][1]
Q0037	Kx(2xRxSIN(A))	[↱][STO] *then* [RCL] *before each alpha input*
Q0038	FS? 2	[↱][×][3][2]
Q0039	Kx(CxSIN(A))	[↱][STO] *then* [RCL] *before each alpha input*
Q0040	FS? 3	[↱][×][3][3]
Q0041	KxCxSIN((D÷2)-A)	[↱][STO] *then* [RCL] *before each alpha input*
Q0042	0	[0]
Q0043	x<>y	[x↔y]
Q0044	STOP	[R/S]
Q0045	SF 10	[↱][×][1][·][0]
Q0046	GTO Q	[↰][XEQ][Q]
Q0047	RTN	[↱][+]

Some of the equations within the programs use symbols representing keys that you may not use often, most commonly, the ^ symbol is input with [y^x], the SQ with [x²] and the SQRT with [√x]. There are also some equations used that are too long to show on just one line within this book's format, and these are indicated by a] and a note indicating that they are all one equation, not two lines of input (the insert to the right shows an example).

```
ASIN((F÷D)x(      ]
SIN(→HR(G))))     ]
```

G0001	LBL G	[↰][+][G]
G0002	CLVARS	[↰][←][2]
G0003	CF 0	[↱][×][2][0]
G0004	CF 1	[↱][×][2][1]
G0005	CF 2	[↱][×][2][2]
G0006	CF 3	[↱][×][2][3]
G0007	CF 4	[↱][×][2][4]
G0008	SF 10	[↱][×][1][·][0]
G0009	TYPE SELECTION	[↱][STO] *then* [RCL] *before each alpha input*
G0010	PSE	[↱][R/S]
G0011	CLx	[↰][←][1]
G0012	S1-S2-S3	[↱][STO] *then* [RCL] *before each alpha input*
G0013	x≠0?	[↱][÷][1]
G0014	SF 1	[↱][×][1][1]
G0015	CLx	[↰][←][1]
G0016	A3-S1-A1	[↱][STO] *then* [RCL] *before each alpha input*
G0017	x≠0?	[↱][÷][1]
G0018	SF 2	[↱][×][1][2]
G0019	CLx	[↰][←][1]
G0020	S1-A1-A2	[↱][STO] *then* [RCL] *before each alpha input*
G0021	x≠0?	[↱][÷][1]
G0022	SF 3	[↱][×][1][3]
G0023	CLx	[↰][←][1]
G0024	S1-A1-S2	[↱][STO] *then* [RCL] *before each alpha input*
G0025	x≠0?	[↱][÷][1]

Continued on next page

Continued from previous page

G0026	SF 4	[↱] [×] [1] [4]
G0027	CLx	[↰] [←] [1]
G0028	S1-S2-A2	[↱] [STO] *then* [RCL] *before each alpha input*
G0029	x≠0?	[↱] [÷] [1]
G0030	SF 0	[↱] [×] [1] [0]
G0031	SF 10	[↱] [×] [1] [.] [0]
G0032	FS? 1	[↱] [×] [3] [1]
G0033	INPUT SIDE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0034	FS? 3	[↱] [×] [3] [3]
G0035	INPUT SIDE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0036	FS? 4	[↱] [×] [3] [4]
G0037	INPUT SIDE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0038	FS? 0	[↱] [×] [3] [0]
G0039	INPUT SIDE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0040	FS? 1	[↱] [×] [3] [1]
G0041	STO D	[STO] [D]
G0042	FS? 3	[↱] [×] [3] [3]
G0043	STO D	[STO] [D]
G0044	FS? 4	[↱] [×] [3] [4]
G0045	STO D	[STO] [D]
G0046	FS? 0	[↱] [×] [3] [0]
G0047	STO D	[STO] [D]
G0048	FS? 2	[↱] [×] [3] [2]
G0049	INPUT ANGLE 3	[↱] [STO] *then* [RCL] *before each alpha input*
G0050	FS? 2	[↱] [×] [3] [2]
G0051	STO I	[STO] [I]
G0052	FS? 1	[↱] [×] [3] [1]
G0053	INPUT SIDE 2	[↱] [STO] *then* [RCL] *before each alpha input*
G0054	FS? 1	[↱] [×] [3] [1]
G0055	STO F	[STO] [F]
G0056	FS? 2	[↱] [×] [3] [2]
G0057	INPUT SIDE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0058	FS? 2	[↱] [×] [3] [2]
G0059	STO D	[STO] [D]
G0060	FS? 3	[↱] [×] [3] [3]
G0061	INPUT ANGLE 1	[↱] [STO] *then* [RCL] *before each alpha input*

Continued next column

G0062	FS? 3	[↱] [×] [3] [3]
G0063	STO E	[STO] [E]
G0064	FS? 4	[↱] [×] [3] [4]
G0065	INPUT ANGLE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0066	FS? 4	[↱] [×] [3] [4]
G0067	STO E	[STO] [E]
G0068	FS? 0	[↱] [×] [3] [0]
G0069	INPUT SIDE 2	[↱] [STO] *then* [RCL] *before each alpha input*
G0070	FS? 0	[↱] [×] [3] [0]
G0071	STO F	[STO] [F]
G0072	FS? 1	[↱] [×] [3] [1]
G0073	INPUT SIDE 3	[↱] [STO] *then* [RCL] *before each alpha input*
G0074	FS? 1	[↱] [×] [3] [1]
G0075	STO H	[STO] [H]
G0076	FS? 2	[↱] [×] [3] [2]
G0077	INPUT ANGLE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0078	FS? 2	[↱] [×] [3] [2]
G0079	STO E	[STO] [E]
G0080	FS? 3	[↱] [×] [3] [3]
G0081	INPUT ANGLE 2	[↱] [STO] *then* [RCL] *before each alpha input*
G0082	FS? 3	[↱] [×] [3] [3]
G0083	STO G	[STO] [G]
G0084	FS? 4	[↱] [×] [3] [4]
G0085	INPUT SIDE 2	[↱] [STO] *then* [RCL] *before each alpha input*
G0086	FS? 4	[↱] [×] [3] [4]
G0087	STO F	[STO] [F]
G0088	FS? 0	[↱] [×] [3] [0]
G0089	INPUT ANGLE 2	[↱] [STO] *then* [RCL] *before each alpha input*
G0090	FS? 0	[↱] [×] [3] [0]
G0091	STO G	[STO] [G]
G0092	CF 10	[↱] [×] [2] [.] [0]
G0093	FS? 3	[↱] [×] [3] [3]
G0094	180-(→HR(E)+→HR(G))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0095	FS? 3	[↱] [×] [3] [3]
G0096	→HMS	[↱] [5]
G0097	FS? 3	[↱] [×] [3] [3]

Continued on next page

Continued from previous page

Step	Instruction	Keys
G0098	STO I	[STO] [I]
G0099	FS? 3	[↱] [×] [3] [3]
G0100	SF 2	[↱] [×] [1] [2]
G0101	FS? 3	[↱] [×] [3] [3]
G0102	CF 3	[↱] [×] [2] [3]
G0103	FS? 0	[↱] [×] [3] [0]
G0104	ASIN((F÷D)x(SIN(→HR(G))))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0105	FS? 0	[↱] [×] [3] [0]
G0106	→HMS	[↱] [5]
G0107	FS? 0	[↱] [×] [3] [0]
G0108	STO I	[STO] [I]
G0109	FS? 0	[↱] [×] [3] [0]
G0110	180-(→HR(I) +→HR(G))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0111	FS? 0	[↱] [×] [3] [0]
G0112	→HMS	[↱] [5]
G0113	FS? 0	[↱] [×] [3] [0]
G0114	STO E	[STO] [E]
G0115	FS? 0	[↱] [×] [3] [0]
G0116	SF 2	[↱] [×] [1] [2]
G0117	FS? 2	[↱] [×] [3] [2]
G0118	180-(→HR(I) +→HR(E))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0119	FS? 2	[↱] [×] [3] [2]
G0120	→HMS	[↱] [5]
G0121	FS? 2	[↱] [×] [3] [2]
G0122	STO G	[STO] [G]
G0123	FS? 2	[↱] [×] [3] [2]
G0124	Dx((SIN(→HR(I)) ÷(SIN(→HR(G)))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0125	FS? 2	[↱] [×] [3] [2]
G0126	STO F	[STO] [F]
G0127	FS? 2	[↱] [×] [3] [2]
G0128	Dx(COS(→HR(I)))	[↱] [STO] *then* [RCL] *before each alpha input*
G0129	FS? 2	[↱] [×] [3] [2]
G0130	Fx(COS(→HR(G)))	[↱] [STO] *then* [RCL] *before each alpha input*
G0131	FS? 2	[↱] [×] [3] [2]
G0132	+	[+]

Continued next column

Step	Instruction	Keys
G0133	FS? 2	[↱] [×] [3] [2]
G0134	STO H	[STO] [H]
G0135	FS? 4	[↱] [×] [3] [4]
G0136	2x(DxF)x(COS(→HR(E)))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0137	FS? 4	[↱] [×] [3] [4]
G0138	STO T	[STO] [T]
G0139	FS? 4	[↱] [×] [3] [4]
G0140	SQRT(SQ(D)+SQ (F)-T)	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0141	FS? 4	[↱] [×] [3] [4]
G0142	STO H	[STO] [H]
G0143	SF 1	[↱] [×] [1] [1]
G0144	SF 4	[↱] [×] [1] [4]
G0145	FS? 1	[↱] [×] [3] [1]
G0146	(D+F+H)÷2	[↱] [STO] *then* [RCL] *before each alpha input*
G0147	FS? 1	[↱] [×] [3] [1]
G0148	STO P	[STO] [P]
G0149	FS? 1	[↱] [×] [3] [1]
G0150	SQRT(((Px(P-F)) ÷(DxH))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0151	FS? 1	[↱] [×] [3] [1]
G0152	STO T	[STO] [T]
G0153	FS? 1	[↱] [×] [3] [1]
G0154	2xACOS(T)	[↱] [STO] *then* [RCL] *before each alpha input*
G0155	FS? 1	[↱] [×] [3] [1]
G0156	→HMS	[↱] [5]
G0157	FS? 1	[↱] [×] [3] [1]
G0158	STO I	[STO] [I]
G0159	FS? 1	[↱] [×] [3] [1]
G0160	ACOS(SQRT((Px(P-D))÷(FxH)))	***input as 1 equation*** [↱] [STO] *then* [RCL] *before each alpha input*
G0161	FS? 1	[↱] [×] [3] [1]
G0162	2	[2]
G0163	FS? 1	[↱] [×] [3] [1]
G0164	x	[×]
G0165	FS? 1	[↱] [×] [3] [1]
G0166	→HMS	[↱] [5]

Continued on next page

Continued from previous page

G0167	FS? 1	[↱] [×] [3] [1]
G0168	STO G	[STO] [G]
G0169	FS? 1	[↱] [×] [3] [1]
G0170	180-(→HR(I)+ →HR(G))	*input as 1 equation* [↱] [STO] *then* [RCL] *before each alpha input*
G0171	FS? 1	[↱] [×] [3] [1]
G0172	→HMS	[↱] [5]
G0173	FS? 1	[↱] [×] [3] [1]
G0174	STO E	[STO] [E]
G0175	CF 1	[↱] [×] [2] [1]
G0176	SF 10	[↱] [×] [1] [.] [0]
G0177	SIDE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0178	RCL D	[RCL] [D]
G0179	STOP	[R/S]
G0180	ANGLE 1	[↱] [STO] *then* [RCL] *before each alpha input*
G0181	RCL E	[RCL] [E]
G0182	STOP	[R/S]
G0183	SIDE 2	[↱] [STO] *then* [RCL] *before each alpha input*
G0184	RCL F	[RCL] [F]
G0185	STOP	[R/S]
G0186	ANGLE 2	[↱] [STO] *then* [RCL] *before each alpha input*
G0187	RCL G	[RCL] [G]
G0188	STOP	[R/S]
G0189	SIDE 3	[↱] [STO] *then* [RCL] *before each alpha input*
G0190	RCL H	[RCL] [H]
G0191	STOP	[R/S]
G0192	ANGLE 3	[↱] [STO] *then* [RCL] *before each alpha input*
G0193	RCL I	[RCL] [I]
G0194	STOP	[R/S]
G0195	CF 10	[↱] [×] [2] [.] [0]
G0196	0.5x(DxHxSIN(→HR(I)))	*input as 1 equation* [↱] [STO] *then* [RCL] *before each alpha input*
G0197	STO A	[STO] [A]
G0198	SF 10	[↱] [×] [1] [.] [0]
G0199	AREA	[↱] [STO] *then* [RCL] *before each alpha input*
G0200	RCL A	[RCL] [A]

Continued next column

G0201	STOP	[R/S]
G0202	FS? 0	[↱] [×] [3] [0]
G0203	XEQ S	[XEQ] [S]
G0204	GTO G	[↰] [XEQ] [G]
G0205	RTN	[↱] [+]

Required sub-routine

S0001	LBL S	[↰] [+] [S]
S0002	2ND SOLUTION	[↱] [STO] *then* [RCL] *before each alpha input*
S0003	CF 10	[↱] [×] [2] [.] [0]
S0004	180-(→HR(I))	[↱] [STO] *then* [RCL] *before each alpha input*
S0005	RCL G	[RCL] [G]
S0006	STO E	[STO] [E]
S0007	R↓	[R↓]
S0008	→HMS	[↱] [5]
S0009	STO G	[STO] [G]
S0010	RCL F	[RCL] [F]
S0011	STO D	[STO] [D]
S0012	180-(→HR(E)+ →HR(G))	*input as 1 equation* [↱] [STO] *then* [RCL] *before each alpha input*
S0013	→HMS	[↱] [5]
S0014	STO I	[STO] [I]
S0015	SIN(→HR(I))	[↱] [STO] *then* [RCL] *before each alpha input*
S0016	SIN(→HR(G))	[↱] [STO] *then* [RCL] *before each alpha input*
S0017	÷	[÷]
S0018	RCLx D	[RCL] [×] [D]
S0019	STO F	[STO] [F]
S0020	Dx(COS(→HR(I)))	[↱] [STO] *then* [RCL] *before each alpha input*
S0021	Fx(COS(→HR(G)))	[↱] [STO] *then* [RCL] *before each alpha input*
S0022	+	[+]
S0023	STO H	[STO] [H]
S0024	SF 10	[↱] [×] [1] [.] [0]
S0025	SIDE 1	[↱] [STO] *then* [RCL] *before each alpha input*
S0026	RCL D	[RCL] [D]
S0027	STOP	[R/S]
S0028	ANGLE 1	[↱] [STO] *then* [RCL] *before each alpha input*

Continued on next page

Continued from previous page

```
S0029  RCL E          [RCL] [E]
S0030  STOP           [R/S]
S0031  SIDE 2         [↱] [STO] then [RCL]
                      before each alpha input
S0032  RCL F          [RCL] [F]
S0033  STOP           [R/S]
S0034  ANGLE 2        [↱] [STO] then [RCL]
                      before each alpha input
S0035  RCL G          [RCL] [G]
S0036  STOP           [R/S]
S0037  SIDE 3         [↱] [STO] then [RCL]
                      before each alpha input
S0038  RCL H          [RCL] [H]
S0039  STOP           [R/S]
S0040  ANGLE 3        [↱] [STO] then [RCL]
                      before each alpha input
S0041  RCL I          [RCL] [I]
S0042  STOP           [R/S]
S0043  CF 10          [↱] [×] [2] [·] [0]
S0044  0.5x(DxHxSIN   ] input as 1 equation
       (→HR(I)))      ] [↱] [STO] then [RCL]
                      before each alpha input
S0045  STO A          [STO] [A]
S0046  SF 10          [↱] [×] [1] [·] [0]
S0047  AREA           [↱] [STO] then [RCL]
                      before each alpha input
S0048  RCL A          [RCL] [A]
S0049  STOP           [R/S]
S0050  RTN            [↱] [+]
```

·Vertical Curves

```
V0001  LBL V          [↰] [+] [V]
V0002  CLVARS         [↰] [←] [2]
V0003  SF 10          [↱] [×] [1] [·] [0]
V0004  CF 1           [↱] [×] [2] [1]
V0005  INPUT BEG STA  [↱] [STO] then [RCL]
                      before each alpha input
V0006  STO S          [STO] [S]
V0007  INPUT BEG ELEV [↱] [STO] then [RCL]
                      before each alpha input
V0008  STO E          [STO] [E]
V0009  GRADE IN:      [↱] [STO] then [RCL]
                      before each alpha input
V0010  STO I          [STO] [I]
V0011  CLx            [↰] [←] [1]
V0012  GRADE OUT:     [↱] [STO] then [RCL]
                      before each alpha input
```

Continued next column

```
V0013  STO O          [STO] [O]
V0014  CLx            [↰] [←] [1]
V0015  INPUT LENGTH   [↱] [STO] then [RCL]
                      before each alpha input
V0016  STO L          [STO] [L]
V0017  x≠0?           [↱] [÷] [1]
V0018  SF 1           [↱] [×] [1] [1]
V0019  RCL S          [RCL] [S]
V0020  STO R          [STO] [R]
V0021  0              [0]
V0022  STO S          [STO] [S]
V0023  FS? 1          [↱] [×] [3] [1]
V0024  GTO Z          [↰] [XEQ] [Z]
Y0001  LBL Y          [↰] [+] [Y]
Y0002  INPUT STA:     [↱] [STO] then [RCL]
                      before each alpha input
Y0003  STO S          [STO] [S]
Y0004  RCL- R         [RCL] [−] [R]
Y0005  RCL I          [RCL] [I]
Y0006  100            [1] [0] [0]
Y0007  ÷              [÷]
Y0008  x              [×]
Y0009  RCL+ E         [RCL] [+] [E]
Y0010  RCL S          [RCL] [S]
Y0011  x<>y           [x↔y]
Y0012  STOP           [R/S]
Y0013  GTO Y          [↰] [XEQ] [Y]
```

— *Required sub-routine* —

```
Z0001  LBL Z          [↰] [+] [Z]
Z0002  RCL L          [RCL] [L]
Z0003  ENTER          [ENTER]
Z0004  RCL O          [RCL] [O]
Z0005  RCL- I         [RCL] [−] [I]
Z0006  x<>y           [x↔y]
Z0007  50             [5] [0]
Z0008  ÷              [÷]
Z0009  ÷              [÷]
Z0010  STO M          [STO] [M]
Z0011  RCL I          [RCL] [I]
Z0012  RCLx L         [RCL] [×] [L]
Z0013  RCL I          [RCL] [I]
Z0014  RCL- O         [RCL] [−] [O]
Z0015  ÷              [÷]
Z0016  ENTER          [ENTER]
Z0017  RCL+ R         [RCL] [+] [R]
Z0018  STO S          [STO] [S]
X0001  LBL X          [↰] [+] [X]
```

Continued on next page

Continued from previous page

X0002 RCL- R	[RCL] [−] [R]
X0003 100	[1] [0] [0]
X0004 ÷	[÷]
X0005 ENTER	[ENTER]
X0006 ENTER	[ENTER]
X0007 RCLx M	[RCL] [×] [M]
X0008 RCL+ I	[RCL] [+] [I]
X0009 x	[×]
X0010 RCL+ E	[RCL] [+] [E]
X0011 RCL S	[RCL] [S]
X0012 x<>y	[x↔y]
X0013 0	[0]
X0014 STO S	[STO] [S]
X0015 R↓	[R↓]
X0016 STOP	[R/S]
X0017 INPUT STA:	[↱] [STO] *then* [RCL] *before each alpha input*
X0018 STO S	[STO] [S]
X0019 GTO X	[↰] [XEQ] [X]
X0020 RTN	[↱] [+]

– Vertical Intersections

P0001 LBL P	[↰] [+] [P]
P0002 CLVARS	[↰] [←] [2]
P0003 SF 10	[↱] [×] [1] [.] [0]
P0004 INPUT STA 1	[↱] [STO] *then* [RCL] *before each alpha input*
P0005 STO T	[STO] [T]
P0006 INPUT ELEV 1	[↱] [STO] *then* [RCL] *before each alpha input*
P0007 STO F	[STO] [F]
P0008 INPUT STA 2	[↱] [STO] *then* [RCL] *before each alpha input*
P0009 RCL- T	[RCL] [−] [T]
P0010 STO M	[STO] [M]
P0011 INPUT ELEV 2	[↱] [STO] *then* [RCL] *before each alpha input*
P0012 RCL F	[RCL] [F]
P0013 x<>y	[x↔y]
P0014 -	[−]
P0015 STO N	[STO] [N]
P0016 GRADE IN:	[↱] [STO] *then* [RCL] *before each alpha input*
P0017 100	[1] [0] [0]
P0018 ÷	[÷]

Continued on next column

P0019 STO I	[STO] [I]
P0020 GRADE OUT:	[↱] [STO] *then* [RCL] *before each alpha input*
P0021 100	[1] [0] [0]
P0022 ÷	[÷]
P0023 STO O	[STO] [O]
P0024 RCL N	[RCL] [N]
P0025 RCL O	[RCL] [O]
P0026 ENTER	[ENTER]
P0027 RCL N	[RCL] [N]
P0028 x<>y	[x↔y]
P0029 ÷	[÷]
P0030 RCL+ M	[RCL] [+] [M]
P0031 RCLx O	[RCL] [×] [O]
P0032 RCL I	[RCL] [I]
P0033 RCL- O	[RCL] [−] [O]
P0034 ÷	[÷]
P0035 +/-	[+/−]
P0036 RCL+ T	[RCL] [+] [T]
P0037 STO X	[STO] [X]
P0038 ENTER	[ENTER]
P0039 RCL- T	[RCL] [−] [T]
P0040 RCLx I	[RCL] [×] [I]
P0041 RCL+ F	[RCL] [+] [F]
P0042 RCL X	[RCL] [X]
P0043 x<>y	[x↔y]
P0044 RTN	[↱] [+]

– Elevation-Find Station

E0001 LBL E	[↰] [+] [E]
E0002 STO H	[STO] [H]
E0003 RCL L	[RCL] [L]
E0004 x=0?	[↱] [÷] [6]
E0005 GTO H	[↰] [XEQ] [H]
E0006 RCL O	[RCL] [O]
E0007 RCL- I	[RCL] [−] [I]
E0008 100	[1] [0] [0]
E0009 ÷	[÷]
E0010 RCL÷ L	[RCL] [÷] [L]
E0011 STO Y	[STO] [Y]
E0012 RCL E	[RCL] [E]
E0013 RCL- H	[RCL] [−] [H]
E0014 STO A	[STO] [A]
E0015 RCL Y	[RCL] [Y]

Continued on next page

Continued from previous page

E0016	2	[2]
E0017	x	[×]
E0018	STO K	[STO] [K]
E0019	RCLx A	[RCL] [×] [A]
E0020	STO K	[STO] [K]
E0021	RCL I	[RCL] [I]
E0022	100	[1] [0] [0]
E0023	÷	[÷]
E0024	STO X	[STO] [X]
E0025	x²	[x²]
E0026	RCL- K	[RCL] [−] [K]
E0027	√x	[√x]
E0028	STO K	[STO] [K]
E0029	RCL X	[RCL] [X]
E0030	+/-	[+/−]
E0031	+	[+]
E0032	RCL÷ Y	[RCL] [÷] [Y]
E0033	RCL+ R	[RCL] [+] [R]
E0034	STO S	[STO] [S]
E0035	RCL X	[RCL] [X]
E0036	+/-	[+/−]
E0037	RCL- K	[RCL] [−] [K]
E0038	RCL÷ Y	[RCL] [÷] [Y]
E0039	RCL+ R	[RCL] [+] [R]
E0040	STO T	[STO] [T]
E0041	RCL S	[RCL] [S]
E0042	x<>y	[x↔y]
E0043	STOP	[R/S]
E0044	FS? 1	[↱] [×] [3] [1]
E0045	GTO Z	[↰] [XEQ] [Z]
E0046	RTN	[↱] [+]

— Required sub-routine

H0001	LBL H	[↰] [+] [H]
H0002	RCL E	[RCL] [E]
H0003	RCL- H	[RCL] [−] [H]
H0004	RCL I	[RCL] [I]
H0005	100	[1] [0] [0]
H0006	÷	[÷]
H0007	÷	[÷]
H0008	+/-	[+/−]
H0009	RCL+ R	[RCL] [+] [R]
H0010	STO S	[STO] [S]
H0011	VIEW S	[↱] [0] [S]
H0012	0	[0]
H0013	STO S	[STO] [S]
H0014	GTO Y	[↰] [XEQ] [Y]
H0015	RTN	[↱] [+]

End Miscellaneous Programming

Coordinate Geometry Programming

– Traverse

Step	Instruction	Keys
T0001	LBL T	[Left-shift] [+] [T]
T0002	CLVARS	[Left-shift] [←] [2]
T0003	CF 2	[Right-shift] [×] [2] [2]
T0004	SF 10	[Right-shift] [×] [1] [·] [0]
T0005	0	[0]
T0006	STO S	[STO] [S]
T0007	STO G	[STO] [G]
T0008	STO M	[STO] [M]
T0009	AZ=0 BRG=1	[Right-shift] [STO] *then* [RCL] *before each alpha input*
T0010	STO Z	[STO] [Z]
T0011	x≠0?	[Right-shift] [÷] [1]
T0012	SF 2	[Right-shift] [×] [1] [2]
T0013	NORTHING:	[Right-shift] [STO] *then* [RCL] *before each alpha input*
T0014	STO Y	[STO] [Y]
T0015	STO N	[STO] [N]
T0016	STO U	[STO] [U]
T0017	EASTING:	[Right-shift] [STO] *then* [RCL] *before each alpha input*
T0018	STO X	[STO] [X]
T0019	STO E	[STO] [E]
T0020	STO O	[STO] [O]
J0001	LBL J	[Left-shift] [+] [J]
J0002	RCL Z	[RCL] [Z]
J0003	x=0?	[Right-shift] [÷] [6]
J0004	AZIMUTH:	[Right-shift] [STO] *then* [RCL] *before each alpha input*
J0005	FS? 2	[Right-shift] [×] [3] [2]
J0006	XEQ B	[XEQ] [B]
J0007	→HR	[Left-shift] [5]
J0008	STO A	[STO] [A]
J0009	DISTANCE:	[Right-shift] [STO] *then* [RCL] *before each alpha input*
J0010	STO D	[STO] [D]
J0011	STO+ S	[STO] [+] [S]
J0012	θ,r→y,x	[Right-shift] [4]
J0013	STO L	[STO] [L]
J0014	STO+ N	[STO] [+] [N]
J0015	x<>y	[x↔y]
J0016	STO O	[STO] [O]
J0017	STO+ E	[STO] [+] [E]
J0018	RCL L	[RCL] [L]
J0019	RCL O	[RCL] [O]
J0020	2	[2]

Continued next column

Step	Instruction	Keys
J0021	÷	[÷]
J0022	RCL+ M	[RCL] [+] [M]
J0023	x	[×]
J0024	+	[+]
J0025	STO+ G	[STO] [+] [G]
J0026	RCL O	[RCL] [O]
J0027	STO+ M	[STO] [+] [M]
J0028	VIEW N	[Right-shift] [0] [N]
J0029	VIEW E	[Right-shift] [0] [E]
J0030	GTO J	[Left-shift] [XEQ] [J]
J0031	RTN	[Right-shift] [+]

– Traverse Closure

Step	Instruction	Keys
K0001	LBL K	[Left-shift] [+] [K]
K0002	RCL E	[RCL] [E]
K0003	RCL- X	[RCL] [−] [X]
K0004	RCL N	[RCL] [N]
K0005	RCL- Y	[RCL] [−] [Y]
K0006	y,x→θ,r	[Left-shift] [4]
K0007	STO D	[STO] [D]
K0008	x<>y	[x↔y]
K0009	180	[1] [8] [0]
K0010	+	[+]
K0011	→HMS	[Right-shift] [5]
K0012	STO A	[STO] [A]
K0013	FS? 2	[Right-shift] [×] [3] [2]
K0014	XEQ N	[XEQ] [N]
K0015	CLOSE ERROR	[Right-shift] [STO] *then* [RCL] *before each alpha input*
K0016	PSE	[Right-shift] [R/S]
K0017	FS? 2	[Right-shift] [×] [3] [2]
K0018	VIEW B	[Right-shift] [0] [B]
K0019	FS? 2	[Right-shift] [×] [3] [2]
K0020	VIEW Q	[Right-shift] [0] [Q]
K0021	RCL Z	[RCL] [Z]
K0022	x=0?	[Right-shift] [÷] [6]
K0023	VIEW A	[Right-shift] [0] [A]
K0024	VIEW D	[Right-shift] [0] [D]
K0025	RCL S	[RCL] [S]
K0026	RCL÷ D	[RCL] [÷] [D]
K0027	STO P	[STO] [P]
K0028	PRCSN RATIO 1:	[Right-shift] [STO] *then* [RCL] *before each alpha input*

Continued on next page

Continued from previous page

K0029	PSE	[↱] [R/S]
K0030	VIEW P	[↱] [0] [P]
K0031	RCL G	[RCL] [G]
K0032	ABS	[↰] [y^x]
K0033	STO A	[STO] [A]
K0034	AREA:	[↱] [STO] *then* [RCL] *before each alpha input*
K0035	PSE	[↱] [R/S]

Continued next column

K0036	VIEW A	[↱] [0] [A]
K0037	SUM H DIST:	[↱] [STO] *then* [RCL] *before each alpha input*
K0038	PSE	[↱] [R/S]
K0039	VIEW S	[↱] [0] [S]
K0040	CF 2	[↱] [×] [2] [2]
K0041	STOP	[R/S]
K0042	RTN	[↱] [+]

Inversing

L0001	LBL L	[↰] [+] [L]
L0002	CF 2	[↱] [×] [2] [2]
L0003	SF 1	[↱] [×] [1] [1]
L0004	FS? 4	[↱] [×] [3] [4]
L0005	CF 1	[↱] [×] [2] [1]
L0006	FS? 3	[↱] [×] [3] [3]
L0007	CF 1	[↱] [×] [2] [1]
L0008	SF 10	[↱] [×] [1] [·] [0]
L0009	AZ=0 BRG=1	[↱] [STO] *then* [RCL] *before each alpha input*
L0010	STO Z	[STO] [Z]
L0011	X≠0?	[↱] [÷] [1]
L0012	SF 2	[↱] [×] [1] [2]
L0013	FS? 4	[↱] [×] [3] [4]
L0014	RCL Y	[RCL] [Y]
L0015	FS? 1	[↱] [×] [3] [1]
L0016	NORTHING:	[↱] [STO] *then* [RCL] *before each alpha input*
L0017	STO H	[STO] [H]
L0018	FS? 4	[↱] [×] [3] [4]
L0019	RCL X	[RCL] [X]
L0020	FS? 1	[↱] [×] [3] [1]
L0021	EASTING:	[↱] [STO] *then* [RCL] *before each alpha input*
L0022	STO I	[STO] [I]
R0001	LBL R	[↰] [+] [R]
R0002	NORTHING:	[↱] [STO] *then* [RCL] *before each alpha input*
R0003	STO J	[STO] [J]
R0004	EASTING:	[↱] [STO] *then* [RCL] *before each alpha input*

Continued next column

R0005	STO K	[STO] [K]
R0006	RCL I	[RCL] [I]
R0007	RCL- K	[RCL] [−] [K]
R0008	RCL H	[RCL] [H]
R0009	RCL- J	[RCL] [−] [J]
R0010	y,x→θ,r	[↰] [4]
R0011	STO D	[STO] [D]
R0012	x<>y	[x↔y]
R0013	180	[1] [8] [0]
R0014	+	[+]
R0015	→HMS	[↱] [5]
R0016	STO A	[STO] [A]
R0017	FS? 2	[↱] [×] [3] [2]
R0018	XEQ N	[XEQ] [N]
R0019	FS? 2	[↱] [×] [3] [2]
R0020	VIEW B	[↱] [0] [B]
R0021	FS? 2	[↱] [×] [3] [2]
R0022	VIEW Q	[↱] [0] [Q]
R0023	RCL Z	[RCL] [Z]
R0024	X=0?	[↱] [÷] [6]
R0025	VIEW A	[↱] [0] [A]
R0026	VIEW D	[↱] [0] [D]
R0027	FS? 4	[↱] [×] [3] [4]
R0028	GTO J	[↰] [XEQ] [J]
R0029	MORE=0 STAKE=1	[↱] [STO] *then* [RCL] *before each alpha input*
R0030	X≠0?	[↱] [÷] [1]
R0031	GTO R	[↰] [XEQ] [R]
R0032	X=0?	[↱] [÷] [6]
R0033	CF 3	[↱] [×] [2] [3]
R0034	CF 10	[↱] [×] [2] [·] [0]
R0035	GTO L	[↰] [XEQ] [L]
R0036	RTN	[↱] [+]

Line	Instruction	Keystrokes
I0001	LBL I	[◄] [+] [I]
I0002	CLVARS	[◄] [←] [2]
I0003	CF 0	[►] [×] [2] [0]
I0004	CF 1	[►] [×] [2] [1]
I0005	CF 2	[►] [×] [2] [2]
I0006	CF 3	[►] [×] [2] [3]
I0007	CF 4	[►] [×] [2] [4]
I0008	SF 10	[►] [×] [1] [·] [0]
I0009	BEG NORTHING:	[►] [STO] *then* [RCL] *before each alpha input*
I0010	STO Y	[STO] [Y]
I0011	BEG EASTING:	[►] [STO] *then* [RCL] *before each alpha input*
I0012	STO X	[STO] [X]
I0013	END NORTHING:	[►] [STO] *then* [RCL] *before each alpha input*
I0014	STO U	[STO] [U]
I0015	END EASTING:	[►] [STO] *then* [RCL] *before each alpha input*
I0016	STO O	[STO] [O]
I0017	RCL- X	[RCL] [−] [X]
I0018	RCL U	[RCL] [U]
I0019	RCL- Y	[RCL] [−] [Y]
I0020	y,x→θ,r	[◄] [4]
I0021	STO S	[STO] [S]
I0022	x<>y	[x↔y]
I0023	STO T	[STO] [T]
I0024	CLx	[◄] [←] [1]
I0025	AZ=0 BRG=1	[►] [STO] *then* [RCL] *before each alpha input*
I0026	x≠0?	[►] [÷] [1]
I0027	SF 2	[►] [×] [1] [2]
I0028	X=0?	[►] [÷] [6]
I0029	SF 6	[►] [×] [1] [6]
I0030	STO Z	[STO] [Z]
I0031	CLx	[◄] [←] [1]
I0032	FS? 6	[►] [×] [3] [6]
I0033	AZ-AZ=1	[►] [STO] *then* [RCL] *before each alpha input*
I0034	FS? 2	[►] [×] [3] [2]
I0035	BRG-BRG=1	[►] [STO] *then* [RCL] *before each alpha input*
I0036	x≠0?	[►] [÷] [1]

Continued next column

Line	Instruction	Keystrokes
I0037	SF 1	[►] [×] [1] [1]
I0038	CLx	[◄] [←] [1]
I0039	FS? 6	[►] [×] [3] [6]
I0040	AZ-DIST=1	[►] [STO] *then* [RCL] *before each alpha input*
I0041	FS? 2	[►] [×] [3] [2]
I0042	BRG-DIST=1	[►] [STO] *then* [RCL] *before each alpha input*
I0043	X≠0?	[►] [÷] [1]
I0044	SF 3	[►] [×] [1] [3]
I0045	CLx	[◄] [←] [1]
I0046	DIST-DIST=1	[►] [STO] *then* [RCL] *before each alpha input*
I0047	X≠0?	[►] [÷] [1]
I0048	SF 4	[►] [×] [1] [4]
I0049	CLx	[◄] [←] [1]
I0050	FS? 6	[►] [×] [3] [6]
I0051	AZ-OS=1	[►] [STO] *then* [RCL] *before each alpha input*
I0052	FS? 2	[►] [×] [3] [2]
I0053	BRG-OS=1	[►] [STO] *then* [RCL] *before each alpha input*
I0054	X≠0?	[►] [÷] [1]
I0055	SF 0	[►] [×] [1] [0]
I0056	CLx	[◄] [←] [1]
I0057	FS? 0	[►] [×] [3] [0]
I0058	XEQ W	[XEQ] [W]
I0059	FS? 1	[►] [×] [3] [1]
I0060	XEQ W	[XEQ] [W]
I0061	FS? 3	[►] [×] [3] [3]
I0062	XEQ W	[XEQ] [W]
I0063	STO C	[STO] [C]
I0064	→HMS	[►] [5]
I0065	STO A	[STO] [A]
I0066	RCL T	[RCL] [T]
I0067	RCL- C	[RCL] [−] [C]
I0068	STO J	[STO] [J]
I0069	90	[9] [0]
I0070	STO H	[STO] [H]
I0071	+	[+]
I0072	180	[1] [8] [0]
I0073	-	[−]
I0074	+/-	[+/-]

Continued on next page

Continued from previous page

I0075 STO K	[STO] [K]
I0076 FS? 4	[↱] [×] [3] [4]
I0077 DISTANCE:	[↱] [STO] *then* [RCL] *before each alpha input*
I0078 FS? 4	[↱] [×] [3] [4]
I0079 STO D	[STO] [D]
I0080 CF 10	[↱] [×] [2] [·] [0]
I0081 FS? 0	[↱] [×] [3] [0]
I0082 G=SxSIN(J)	[↱] [STO] *then* [RCL] *before each alpha input*
I0083 FS? 0	[↱] [×] [3] [0]
I0084 STO G	[STO] [G]
I0085 SF 10	[↱] [×] [1] [·] [0]
I0086 FS? 1	[↱] [×] [3] [1]
I0087 XEQ W	[XEQ] [W]
I0088 FS? 1	[↱] [×] [3] [1]
I0089 STO P	[STO] [P]
I0090 FS? 3	[↱] [×] [3] [3]
I0091 DISTANCE:	[↱] [STO] *then* [RCL] *before each alpha input*
I0092 FS? 3	[↱] [×] [3] [3]
I0093 STO G	[STO] [G]
I0094 FS? 4	[↱] [×] [3] [4]
I0095 DISTANCE:	[↱] [STO] *then* [RCL] *before each alpha input*
I0096 FS? 4	[↱] [×] [3] [4]
I0097 STO G	[STO] [G]
I0098 CF 10	[↱] [×] [2] [·] [0]
I0099 FS? 0	[↱] [×] [3] [0]
I0100 SxSIN(J)	[↱] [STO] *then* [RCL] *before each alpha input*
I0101 FS? 0	[↱] [×] [3] [0]
I0102 STO G	[STO] [G]
I0103 FS? 0	[↱] [×] [3] [0]
I0104 SxCOS(J)	[↱] [STO] *then* [RCL] *before each alpha input*
I0105 FS? 0	[↱] [×] [3] [0]
I0106 STO D	[STO] [D]
I0107 FS? 1	[↱] [×] [3] [1]
I0108 P-T	[↱] [STO] *then* [RCL] *before each alpha input*
I0109 FS? 1	[↱] [×] [3] [1]
I0110 STO K	[STO] [K]
I0111 FS? 1	[↱] [×] [3] [1]
I0112 180-(K+J)	[↱] [STO] *then* [RCL] *before each alpha input*

Continued next column

I0113 FS? 1	[↱] [×] [3] [1]
I0114 STO H	[STO] [H]
I0115 FS? 1	[↱] [×] [3] [1]
I0116 (SxSIN(K))÷SIN(H)	[↱] [STO] *then* [RCL] *before each alpha input*
I0117 FS? 1	[↱] [×] [3] [1]
I0118 STO D	[STO] [D]
I0119 FS? 3	[↱] [×] [3] [3]
I0120 T-C	[↱] [STO] *then* [RCL] *before each alpha input*
I0121 FS? 3	[↱] [×] [3] [3]
I0122 STO J	[STO] [J]
I0123 FS? 3	[↱] [×] [3] [3]
I0124 (SxSIN(J))^2	[↱] [STO] *then* [RCL] *before each alpha input*
I0125 FS? 3	[↱] [×] [3] [3]
I0126 STO M	[STO] [M]
I0127 FS? 3	[↱] [×] [3] [3]
I0128 SQ(G)	[↱] [STO] *then* [RCL] *before each alpha input*
I0129 FS? 3	[↱] [×] [3] [3]
I0130 STO L	[STO] [L]
I0131 FS? 3	[↱] [×] [3] [3]
I0132 SxCOS(J)	[↱] [STO] *then* [RCL] *before each alpha input*
I0133 FS? 3	[↱] [×] [3] [3]
I0134 SQRT(L-M)	[↱] [STO] *then* [RCL] *before each alpha input*
I0135 FS? 3	[↱] [×] [3] [3]
I0136 +	[+]
I0137 FS? 3	[↱] [×] [3] [3]
I0138 STO D	[STO] [D]
I0139 FS? 4	[↱] [×] [3] [4]
I0140 SQ(S)+SQ(D)-SQ(G)	[↱] [STO] *then* [RCL] *before each alpha input*
I0141 FS? 4	[↱] [×] [3] [4]
I0142 2xSxD	[↱] [STO] *then* [RCL] *before each alpha input*
I0143 FS? 4	[↱] [×] [3] [4]
I0144 ÷	[÷]
I0145 FS? 4	[↱] [×] [3] [4]
I0146 ACOS	[↰] [COS]
I0147 FS? 4	[↱] [×] [3] [4]
I0148 STO J	[STO] [J]
I0149 FS? 4	[↱] [×] [3] [4]

Continued on next page

Continued from previous page

Step	Instruction	Keys
I0150	RCL T	[RCL] [T]
I0151	FS? 4	[↱] [×] [3] [4]
I0152	RCL- J	[RCL] [−] [J]
I0153	FS? 4	[↱] [×] [3] [4]
I0154	STO C	[STO] [C]
I0155	FS? 4	[↱] [×] [3] [4]
I0156	→HMS	[↱] [5]
I0157	FS? 4	[↱] [×] [3] [4]
I0158	STO A	[STO] [A]
I0159	RCL C	[RCL] [C]
I0160	RCL D	[RCL] [D]
I0161	θ,r→y,x	[↱] [4]
I0162	RCL+ Y	[RCL] [+] [Y]
I0163	STO N	[STO] [N]
I0164	x<>y	[x↔y]
I0165	RCL+ X	[RCL] [+] [X]
I0166	STO E	[STO] [E]
I0167	XEQ M	[XEQ] [M]
I0168	VIEW D	[↱] [0] [D]
I0169	VIEW N	[↱] [0] [N]
I0170	VIEW E	[↱] [0] [E]
I0171	RCL E	[RCL] [E]
I0172	RCL- O	[RCL] [−] [O]
I0173	RCL N	[RCL] [N]
I0174	RCL- U	[RCL] [−] [U]
I0175	y,x→θ,r	[↰] [4]
I0176	STO D	[STO] [D]
I0177	x<>y	[x↔y]
I0178	180	[1] [8] [0]
I0179	+	[+]
I0180	→HMS	[↱] [5]
I0181	STO A	[STO] [A]
I0182	XEQ M	[XEQ] [M]
I0183	VIEW D	[↱] [0] [D]
I0184	CF 6	[↱] [×] [2] [6]
I0185	GTO I	[↰] [XEQ] [I]
I0186	RTN	[↱] [+]

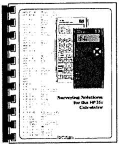
— Required sub-routine

Step	Instruction	Keys
W0001	LBL W	[↰] [+] [W]
W0002	RCL Z	[RCL] [Z]
W0003	X=0?	[↱] [÷] [6]
W0004	AZIMUTH:	[↱] [STO] *then* [RCL] *before each alpha input*
W0005	FS? 2	[↱] [×] [3] [2]
W0006	XEQ B	[XEQ] [B]
W0007	→HR	[↰] [5]
W0008	RTN	[↱] [+]

— Required sub-routine

Step	Instruction	Keys
M0001	LBL M	[↰] [+] [M]
M0002	FS? 6	[↱] [×] [3] [6]
M0003	VIEW A	[↱] [0] [A]
M0004	FS? 2	[↱] [×] [3] [2]
M0005	RCL A	[RCL] [A]
M0006	FS? 2	[↱] [×] [3] [2]
M0007	XEQ N	[XEQ] [N]
M0008	FS? 2	[↱] [×] [3] [2]
M0009	VIEW B	[↱] [0] [B]
M0010	FS? 2	[↱] [×] [3] [2]
M0011	VIEW Q	[↱] [0] [Q]
M0012	RTN	[↱] [+]

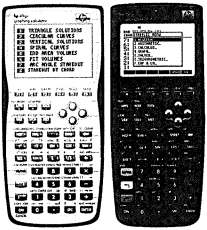
End Coordinate Geometry Programming



New HP35s Solutions Book. All of the same programs that are in this hp33s book, but it can also store points!! Allows storage and recall of up to 800 coordinates pairs .

The HP35s calculator allows direct addressing and that eliminates several subroutines, leaving a bunch of labels available for future programs. We will have free downloads (PDFs) of additional programming for this calculator on our website (www.SoftwareByDZign,com) soon. *Stock # 3511*

SOFTWARE FOR THE hp 49g+ and HP50g CALCULATORS



- All of the products are on a standard 64 Megabyte SD card, leaving approximately 62 Megabytes of backup/storage memory for the user. All angular addition, subtraction, multiplication and division are done directly in Degrees, minutes and seconds format.
- D'Zign handles all of the software support directly and updates are available to users as a basic service from D'Zign.
- **NOTE!** HP49G+ calculators will need to have the operating system updated (instructions and software included with our products) The currently available products are all updated Ver. 2 editions.

hp 49g+/HP50g Basic Cogo *Pac* *Stock # 5031*

Our basic pac, designed for students taking the required course in Coordinate Geometry, is also perfect for doing the everyday office calculations for surveying. In addition to ALL of the normal cogo programming, including pre-determined areas, intersections, and rotation/transformation, it solves vertical, circular and spiral curve problems with just a few keystrokes. PC transfer/translation programming is also available to users (Convertor, *stock #5050*).

hp 49g+/HP50g *Surveying Pac* *Stock # 5081*

All of the programming of the hp49g+ Student Pac (above), plus Field Stakeout solutions, curve through three points, curve common to three tangents, differential, trig and three-wire leveling, 2– and 3-point resection and more. A special conversion program does calculations and/or conversions in Foot-Inch-Fraction or metric units directly, output is in any other format selected. PC transfer/translation programming is also available to users (Convertor, *stock #5050*).

hp 49g+/HP50g *Transportation Pac (TRANSpac)* *Stock # 5083*

All of the programming of the two hp49g+ Pacs above, plus the most powerful and functional Alignment/Offset program ever designed. Handles tangents, both circular and spiral curves, angle points and equation stations. Another program takes the hard work out of Slope Staking by allowing the user to slope-stake large areas from one setup point, calculating all of the information for the slopestake and the reference point. PC transfer/translation programming is also available to users (Convertor, *stock #5050*).

$42.00 U.S.