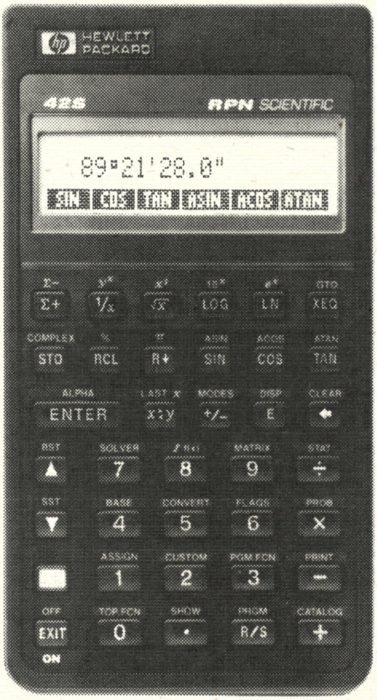


HP42S Surveying Solutions

REVISED, 2ND EDITION

D'Zign

NOTICE

HP42S SURVEYING SOLUTIONS

REVISED 2nd EDITION

published by D'Zign Land Survey & Development
Tollhouse, California, U.S.A. 1993

ISBN 0-944889-14-X

TECHNICAL ASSISTANCE

The program material, instructions and procedures contained in this book assume that the user has a working knowledge of both surveying *and* the general operation of the HP-42S calculator.

Technical assistance is limited to verification of the results shown in the various examples used in this book.

If you have any questions or suggestions regarding this book, or other **D'Zign** publications, please feel free to call us. The number is (209) 297-8025, and someone is available to answer technical questions between the hours of ***8:00 A.M. and Noon***, (Pacific Time Zone), ***Monday through Thursday***.

Before calling for help, take a look through *"The Most Commonly Asked Questions"*, on the inside of the back cover.

WARNING

NOTICE

Table of Contents

INTRODUCTION

Hewlett-Packard has produced a really powerful calculator at a very good price, the HP-42 Scientific Calculator, which lends itself nicely to solving surveying problems. It can not be programmed by insertion of a module, or with a card reader, like the HP-41 series, but it has a really simple system for typing in a program.

the operations index

To find a function for the first time, HP has provided an "Operations Index" on pages 310 through 335 of the instruction manual, which tells you exactly what keystrokes to use to type in the function you want.

Even better, this index gives you the page number that you can refer to if you want to know more about the function you are using. If, while typing in a program, you aren't sure how to input a particular function, simply refer to the Operations Index.

the softkey menus

All of the programs in this book take advantage of the "softkey" menu system built into this calculator. When you want to start a program you stroke **XEQ** and then the softkey corresponding to the program you want, from the menu displayed in the bottom half of the screen.

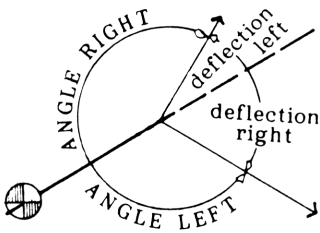
the programs

The purpose of this book is to provide the user with a good surveying program for everyday use in solving field and office problems. When we began programming for this book we decided to use a point-storage system, since it is a real timesaver, particularly in the field. We have designed a traverse, inverse and sideshot program for traversing that is inter-related to the intersection program and the curve solutions program, so that you can use them as a more flexible system.

This also means a longer program, so we suggest that you approach the programming in stages. You are less apt to have programming errors if you don't try to do the whole job at one sitting.

CONVENTIONS

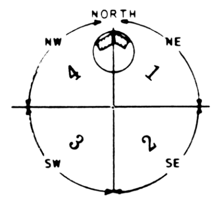
These programs do not make any distinction between "bearing-azimuth" and "field angle" traversing. Once you are in the TRAVERSE program, you may use any method of traversing which you want to use.



You may traverse from one point to another by input of the bearing and quadrant code (or azimuth), or you may turn an angle to the next point.

The illustration to the left shows the relationship of these angles, referenced to the backsight and foresight.

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 HP's came out.



quadrant codes

The bearing is input AND ENTERED, the quadrant code is then input, and a BRG key is stroked.

And, some CAUTIONS: These programs do not recognize sideshots as different from traverse shots. If you use the angle-adjustment, or either the compass or transit adjustment programs, you will get incorrect answers if the traverse contains sideshots. There are ways to keep this from being a problem.

1. Use point numbers for the sideshots that are high enough to be outside the point numbers used for the traverse itself. If you are doing a traverse that will use point numbers 1 through 7, use numbers 11 and up for the sideshots.

2. Calculate the traverse without the sideshots, adjust it, and then do an INVERSE traverse from point to point to point, setting the sideshots as you go. This is quick and simple, because the coordinates are stored by point number in all of these programs.

TRIG

You are probably anxious to start programming the traverse program to try it out, but we urge you to start with this program, and follow the sequence we have used in this book. This first program will give you some practice in programming your calculator, and does a job for you that you will find handy later.

There are always times, while calculating a traverse, that you need to work a quick trig problem before you can do the next step ... like calculating Δ/2 to enter the "curve" routine.

If you do that with this calculator you will have lost your place in the program you were running. Later, this program will be related to the traverse program so that it can be called up without losing your place.

degrees, minutes and seconds

Even better, you can perform all of the trig operations without converting to decimal first. This program does the converting for you, and all input and output is in degrees, minutes and seconds.

getting started

We'll do this first program in two stages, which will also give you some **editing** practice. Begin by stroking the shift key, then the **XEQ** key. The display will show a menu which will be blank (if you haven't yet input any programs) except for **.END.** on the left. The keys just below each of the menu portions will correspond to the menu instruction above it. Stroke the key just below the **.END.** in the display.

Next, stroke the shifted **R/S** key and the display should be as shown to the right.

```
00▸{ 0-Byte Prgm }
01 .END.
```

Begin typing in the program steps shown on the next page. The first step, **LBL "TRIG"**, will replace ".END." as step 01. Each of the following steps will be assigned a step number by the calculator as you type in the program steps.

Steps 03, 05, 07, 09, 11 and 13 are typed while in **alpha** mode. All of the other functions can be accessed through the keyboard and menus. Consult the Operations Index in your owner's manual if you are not sure which keys to stroke.

```
 01▸LBL "TRIG"
 02 SF 21
 03 "SIN"
*04 KEY 1 GTO 01
 05 "COS"
 06 KEY 2 GTO 02
 07 "TAN"
 08 KEY 3 GTO 03
 09 "ASIN"
 10 KEY 4 GTO 04
 11 "ACOS"
 12 KEY 5 GTO 05
 13 "ATAN"
 14 KEY 6 GTO 06
 15 MENU
 16 STOP
 17▸LBL 01
 18 →HR
 19 SIN
 20 RTN
 21▸LBL 02
 22 →HR
 23 COS
 24 RTN
 25▸LBL 03
 26 →HR
 27 TAN
 28 RTN
 29▸LBL 04
 30 ASIN
 31 →HMS
 32 RTN
 33▸LBL 05
 34 ACOS
 35 →HMS
 36 RTN
 37▸LBL 06
 38 ATAN
 39 →HMS
 40 RTN
 41 END
```

When all of the steps have been typed in, go back and proof read the program. All of the steps should be the same as shown in the program listing above. If any of them are not, delete the step and re-type it.

try it out

If you stroke the **XEQ** button now, your display should show a menu with "TRIG" as one of the program options (the only option if you didn't already have any programs stored). Stroking the key under "TRIG" will bring up a menu of SIN COS TAN ASIN ACOS ATAN.

If you use the key which corresponds to these menu labels, instead of those on the keyboard, the trig functions work in degrees, minutes and seconds instead of decimal degrees. Try working a few.

*To access the function, stroke [ ] [PGM.FCN] [▲] [KEYG]

Next, a short editing session. The program, "**TRIG**", works fine as it is, but we want to expand it to do more.

editing

Editing is done "from the bottom, up". If you go into the program and insert or delete any steps it changes the step number of all of the steps below the step which was added or deleted. If you start at the bottom and work upward you will still be able to find your place.

Stroke the shifted **XEQ** key (**GTO**) and then stroke the key under "TRIG" to bring the program to the top of the program memory. Next, stroke the shifted **R/S** (**PRGM**) key to go into **program** mode. You can go to any step in the program by stroking **GTO .**, followed by the step number (4-digit input is required). Let's start by going to step 16, where a step is to be added.
Stroke

[ ] [XEQ] [.] [0] [0] [1] [6]

to get there. Insert the step **GTO 98** between steps 16 and 17.

```
             16 STOP
17 GTO 98 ─────────
             17▸LBL 01
```

Scroll upwards until the program pointer is at **15 MENU**, and then insert the step **LBL 98**.

```
             15 MENU
16▸LBL 98 ─────────
             16 STOP
```

```
14 KEY 6 GTO 06
15 MENU     15 KEY 7 GTO B
            16 KEY 8 GTO B
            17 KEY 9 GTO 99
```

Scroll upwards again, to program step 14, and insert the three new steps shown to the left.

Either scroll upwards or stroke GTO .0001, to put the program pointer at step 01 LBL "TRIG", and add the step, **LBL A**.

```
           01▸LBL "TRIG"
02▸LBL A ─────────────
           02 SF 21
```

With the additions to the program that have been made, the END that was originally program step 41 is now at step 47. There are still some more program steps to be added to our program, at the bottom.

Go to program step 46, which is just above the END, and add the steps shown below.

```
47▶LBL B
48 "DMS+"
49 KEY 1 GTO 07
50 "DMS-"
51 KEY 2 GTO 08
52 "DMSx"
53 KEY 3 GTO 09
54 "DMS÷"
55 KEY 4 GTO 10
56 "D→R"
57 KEY 5 GTO 11
58 "R→D"
59 KEY 6 GTO 12
60 KEY 7 GTO A
61 KEY 8 GTO A
62 KEY 9 GTO 99
63▶LBL 21
64 STOP
65 GTO 21
66▶LBL 99
67 CLMENU
68 EXITALL
69 RTN
70▶LBL 07
71 HMS+
72 RTN
73▶LBL 08
74 HMS-
75 RTN
76▶LBL 09
77 X<>Y
78 →HR
79 x
80 →HMS
81 RTN
82▶LBL 10
83 X<>Y
84 →HR
85 X<>Y
86 ÷
87 →HMS
88 RTN
89▶LBL 11
90 →HR
91 →RAD
92 RTN
93▶LBL 12
94 →DEG
95 →HMS
96 RTN
```

Don't forget to proof read the program when you've finished the programming. A minor error at this stage could go unnoticed now, and be giving wrong answers later.

examples

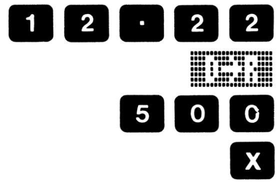
Try out the new routines with this one: Add 25°15'30" to 13°40'20", subtract 6°15'14", multiply by 3 and then divide by two. The keystrokes are shown below.

first angle entered	2 5 . 1 5 3 ENTER
second angle added	1 3 . 4 0 2 
third angle subtracted	6 . 1 5 1 4 
multiply by 3	3 
divide by 2	2 

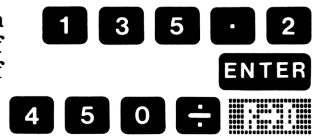
(answer: 49°00'54")

To try out the other two functions, **D→R** and **R→D**, try the following:

1. Find the length of a curve which has a radius of 500.00' and a central angle of 12°22' (length = the radius times the central angle, in radians). (answer: 107.9195')



2. Find the central angle of a curve with an arc length of 135.20' and a radius of 450.00'. (answer: 17°12'51")



The trig functions are much easier to use through the program than from the keyboard, and as an additional plus, you will be able to call this program up as a subroutine from other programs. It does happen that you are half-way through a calculation for something else and need to use one of these functions.

fancy output

After you have input the program, "**DMS**", on page 12 you may want to back up and edit "**TRIG**". You can have the output read in degrees, minutes and seconds (to the nearest tenth second) by using the subroutine, "**DMS**".

Add the three steps, **CLA**, **XEQ** "**DMS**" and **AVIEW** to each of the places in the program that produces an angle as output.

Working from the bottom, upward, this would be between steps 95 and 96, 87 and 88, 80 and 81, 74 and 75, 71 and 72, 39 and 40, 35 and 36, 31 and 32.

This change only costs about 56 bytes of memory, but the enhanced output is clearly an angle, and there is less chance of transposing the numbers when using this form.

CURVES

This program is a CURVE SOLUTION program. It may be used as a "stand alone" program for solving circular curves, and is also called up as a subroutine from the traverse program.

the program listing:

```
01▶LBL "CURVE"
02 CLX
03 CF 05
04 SF 21
05 CF 90
06 CF 91
07 "DELTA"
08 KEY 1 GTO 41
09 "R"
10 KEY 2 GTO 42
11 "L"
12 KEY 3 GTO 43
13 "T"
14 KEY 4 GTO 44
15 "C"
16 KEY 5 GTO 45
17 "D"
18 KEY 6 GTO 46
19 MENU
20 STOP
21▶LBL 41
22 SF 90
23 →HR
24 X<0?
25 SF 05
26 FS? 05
27 +/-
28 2
29 ÷
30 STO 12
31 FS? 91
32 GTO 02
33 STOP
34▶LBL 46
35 →HR
36 100
37 X<>Y
38 ÷
39 →DEG
40▶LBL 42
41 X<0?
42 SF 03
43 ABS
44 STO 11
45 SF 91
46 FS? 90
47 GTO 02
48 STOP
49▶LBL 05
50 FC? 91
51 GTO 07
52▶LBL 43
53 →DEG
54 FS? 91
55 GTO 03
56 2
57 RCL× 12
58 ÷
59 STO 11
60 GTO 02
61▶LBL 03
62 RCL÷ 11
63 2
64 ÷
65 STO 12
66 GTO 02
67▶LBL 45
68 2
69 ÷
70 FS? 91
71 GTO 03
72 RCL 12
73 SIN
74 ÷
75 STO 11
76 GTO 02
77▶LBL 03
78 RCL÷ 11
```

```
79 ASIN
80 STO 12
81 GTO 02
82▸LBL 44
83 FS? 91
84 GTO 03
85 RCL 12
86 TAN
87 ÷
88 STO 11
89 GTO 02
90▸LBL 03
91 RCL÷ 11
92 ATAN
93 STO 12
94▸LBL 02
95 RCL 11
96 "    R = "
97 XEQ 04
98 100
99 X<>Y
100 ÷
101 →DEG
102 GTO 09
103▸LBL 08
104 50
105 X<>Y
106 ÷
107 ASIN
108 2
109 x
110▸LBL 09
111 →HMS
112 ⊦"    D= "
113 XEQ "DMS"
114 AVIEW
115 ADV
116 2
117 RCLx 12
118 FS? 05
119 +/-
120 →HMS
121 "DELTA= "
122 XEQ "DMS"
*123 ⊦"Ч"
124▸LBL 47
125 2
126 RCLx 12
127 RCLx 11
128 →RAD
129 STO 24
130 ⊦"    L= "
131 ARCL ST X
132 AVIEW
133 RCL 12
134 TAN
135 RCLx 11
136 STO 20
137 "    T= "
138 ARCL ST X
139 RCL 12
140 SIN
141 RCLx 11
142 2
143 x
144 STO 18
145 ⊦"Ч    C= "
146 ARCL ST X
147 AVIEW
148 RCL 11
149 ENTER
150 x
151 RCLx 12
152 →RAD
153 STO 19
154 ENTER
155 ENTER
156 RCL 12
157 RCL 11
158 →REC
159 x
160 -
161 STO 22
162 FS? 82
163 GTO 02
164 CF 03
165 1
166 RCL 12
167 COS
168 -
169 RCLx 11
170 "    M= "
171 ARCL ST X
```

* ALPHA ▼ ▼

```
172 RCL 12
173 2
174 ÷
175 TAN
176 RCL× 20
177 ├"␊  E= "
178 ARCL ST X
179 AVIEW
180 STOP
181 RCL 22
182 "Segment"
183 XEQ 01
184 RCL 19
185 "Sector"
186 XEQ 01
187 RCL 12
188 TAN
189 RCL 11
190 X↑2
191 ×
192 RCL 19
193 -
194 "Fillet"
195 CF 90
196 CF 91
197▶LBL 01
198 ADV
199 ├"=␊"
200 ARCL ST X
201 ├" sq ft"
202 AVIEW
203 RTN
204▶LBL 02
205 FS?C 03
206 +/-
207 STO+ 05
208 RCL 18
209 CLMENU
210 GTO "TRAV"
211 RTN
212▶LBL 04
213 ARCL ST X
214 ├"␊"
215 END
```

Once the program is typed in (and proof-read) it is automatically added to the catalogue. Stroking **XEQ** will bring up the program menu, stroking the key which corresponds to **CURVE** will start up the program, showing a menu of DELTA R L T C D .

One of the following, the **central angle** (delta), the **radius** or the **degree of curve**, must be known. Any one of these AND one other part of the curve data are all that are needed to solve any circular curve.

The output, regardless of which parts are known, will be as shown to the right.

Before you can try it out, there is a little more work to do. The program uses the subroutine, "**DMS**", which is part of the next group to be input.

```
    R = 500.0000
   D= 11°27'33.0"
DELTA= 25°00'00.0"
    L= 218.1662
    T= 110.8473
    C= 216.4396
    M= 11.8520
    E= 12.1398
                  R/S
Segment=
1,714.2564 sq ft

Sector=
54,541.5391 sq ft

Fillet=
882.1265 sq ft
```

These subroutines go into a different part of program memory, and include the last one needed by "**CURVE**".

Stroking [ ] [XEQ] (GTO) brings up the program menu.

Next, stroke [TRIG] [ ] [R/S],
(PRGM) and the display will be as shown to the right.

```
00 { xxx-Byte Prgm }
01▶LBL "TRIG"
```

Scroll upwards with [▲] to position the program pointer at 00, and begin typing in the following programs:

```
 01▶LBL "C↑"
 02 FS? 81
 03 XEQ 08
 04 CF 29
 05 FIX 00
 06 CLA
 07 35
 08 XTOA
 09 CLX
 10 R↓
 11 ARCL 13
 12 ├" "
 13 ALENG
*14 FIX 04
 15 SF 29
 16 ├"N= "
 17 ARCL ST Z
 18 ├"␊"
 19 XEQ IND ST X
 20 ├"E= "
 21 ARCL ST Y
 22 AVIEW
 23 R↓
 24 ADV
 25 RTN
 26▶LBL 03
 27 ├"    "
 28 RTN
 29▶LBL 04
 30 ├"      "
 31 RTN
 32▶LBL 05
 33 ├"        "
 34 RTN
 35▶LBL 06
 36 ├"          "
 37 RTN
 38▶LBL 07
 39 ├"            "
 40 RTN
 41▶LBL 08
 42 CLA
 43 " SIDESHOT"
 44 FS? 55
 45 PRA
 46 CLA
 47 RTN
 48▶LBL "DMS"
 49 FS?C 19
 50 CLA
 51 ENTER
 52 STO 19
 53 IP
 54 CF 29
 55 FIX 00
 56 ARCL ST X
 57 ├"°"
 58 -
 59 100
 60 X
 61 ABS
 62 STO 18
 63 IP
 64 XEQ 01
 65 ARCL ST X
 66 ├"'"
 67 RCL 18
 68 FP
 69 100
 70 X
 71 FIX 02
 72 RND
 73 FIX 01
 74 XEQ 01
 75 ARCL ST X
 76 ├"""
 77 CLX
 78 FIX 04
 79 SF 29
 80 RCL 19
 81 RTN
 82▶LBL 01
 83 10
 84 X<>Y
 85 X<Y?
 86 ├"0"
 87 RTN
 88▶LBL "CL"
 89 ΣREG 00
 90 CLΣ
 91 ΣREG 11
 92 CLΣ
 93 0
 94 STO 24
 95 RTN
```

keystroke tips

Step 57 appends a degree symbol. This is input from the alpha keyboard, through "MATH".

Step 66 and step 76 append the minute and second symbols, respectively, and are both input through the "PUNC" key.

*May be varied to suit user's needs.

Information on these keystrokes can be found in the HP42's User's Manual on pages 295 and 296.

There are some program steps which are just spaces, or spaces appended to what is already in the alpha register. For instance, step 12 **appends** one space, step 27 **appends** 3 spaces and step 30 **appends** 4 spaces.

The addition of the spaces and the "line feed" steps is part of what makes the output read correctly whether there is a printer in use or not.

Step 33 **appends** 5 spaces, step 36 **appends** 6 spaces, and step 39 **appends** 7 spaces. When you proof-read the program it is important to check these steps, even though "there isn't anything there".

So far the subroutines that have been input are two that handle display (**C↑** and **DMS**), and one (**CL**) which clears registers 00 through 24, which are not used for coordinate storage by the other programs. This next group manipulates angles, output and prompts. Go to the end of "**CL**" (step 95) and type in the following:

```
96▸LBL "I>"
97 CF 29
98 FIX 00
99 CLA
100 "INV "
101 ARCL 13
102 ⊢"-"
103 ARCL ST X
104 AVIEW
105 RTN
106▸LBL "YN"
107 "YES"
108 KEY 1 GTO 01
109 "NO"
110 KEY 2 GTO 02
111 MENU
112 STOP
113▸LBL 01
114 SF 10
115 GTO 03
116▸LBL 02
117 CF 10
118▸LBL 03
119 CLMENU
*120 EXITALL
121 RTN
122▸LBL "A0"
123 "AZ"
124 SF 85
125 XEQ 06
126 ⊢" = "
127 FIX 04
128 SF 29
129 FS? 10
130 ⊢"?"
131 FS? 85
132 XEQ "DMS"
133 FS?C 85
134 RTN
135 FC? 10
136 ARCL ST X
137 FC? 10
138 RTN
139 PROMPT
140 RTN
```

*Stroke XEQ ENTER , type in EXITALL and stroke ENTER.

```
141▶LBL "A1"
142 FS? 89
143 XEQ 17
144 X<>Y
145 →HR
146 X<>Y
147 ENTER
148 ENTER
149 2
150 ÷
151 IP
152 180
153 x
154 X<>Y
155 LASTX
156 x
157 COS
158 R↑
159 x
160 -
161 FS? 10
162 RTN
163 →HMS
164 GTO "A0"
165 "BRG"
166 XEQ "A0"
167 →HR
168▶LBL "B1"
169 -180
170 X<>Y
171 FS? 89
172 +
173 360
174 MOD
175 ENTER
176 SIN
177 ASIN
178 X<0?
179 +/-
180 →HMS
181 X<>Y
182 90
183 ÷
184 1
185 +
186 IP
187 STO 21
188 GTO IND ST X
189▶LBL 01
190 "N "
191 XEQ 05
192 ⊢" E"
193 RTN
194▶LBL 02
195 "S "
196 XEQ 05
197 ⊢" E"
198 RTN
199▶LBL 03
200 "S "
201 XEQ 05
202 ⊢" W"
203 RTN
204▶LBL 04
205 "N "
206 XEQ 05
207 ⊢" W"
208 RTN
209▶LBL 05
210 STO 21
211 R↓
212 STO 20
213 XEQ "DMS"
214 RTN
215▶LBL 06
216 X>0?
217 RTN
218 360
219 +
220 RTN
221▶LBL 17
222 2
223 -
224 X>0?
225 RTN
226 4
227 +
228 RTN
*229▶LBL "F…"
230 0.013
231▶LBL "…F"
232 CF IND ST X
233 ISG ST X
234 GTO "…F"
235 END
```

* ALPHA ▼ ▼ …

Before beginning this last short group of subroutines, stroke the shifted **XEQ** key (GTO), and then the decimal point twice. This will automatically put an end on the programs so far and give you a new starting point. When you enter the program mode you will again see the display

```
00▶{ 0-Byte Prgm }
01 .END.
```

Scroll upward once, so that the program pointer is at 00, and type in the program steps to the right.

```
01▶LBL "OUT"
02 RCL 13
03 2
04 x
05 24
06 +
07 ENTER
08 ENTER
09 1
10 -
11 RCL IND ST X
12 RCL IND ST Z
13 RTN
14▶LBL "IN"
15 RCL 13
16 2
17 x
18 24
19 +
20 X<>Y
21 STO IND ST Y
22 R↓
23 1
24 -
25 X<>Y
26 STO IND ST Y
27 RTN
28▶LBL "PIN"
29 STO 13
30 XEQ "PN"
31 GTO "IN"
32▶LBL "POUT"
33 STO 13
34 GTO "OUT"
35 END
```

pout

This program is "**Point OUT**" and is used to recall stored coordinates. If you input the point number of a **stored** pair of coordinates and execute "**POUT**" the coordinates will be recalled to the display.

After recall the north coordinate is in the y-register and the east coordinate is in the x-register.

pin

This one stands for "**Point IN**". If you input a point number and execute "**PIN**" you will be prompted for the north coordinate with **N** =?. Input the north coordinate and stroke **R/S** and you will be prompted **E** =?. Input the east coordinate and stroke **R/S**. The coordinates are immediately stored under that point number.

After the subroutines at the top of page 17 have been put in, you can try some coordinate storage. Each pair uses two storage registers, or 18 bytes.

sizing

The default size for the number of storage registers available is 25. This means that registers 00 through 24 may be used for storage. For right now, you need to resize to 50, so that you will have some registers available for storing coordinates. (see page 87)

The **SIZE** function is found in the second menu of "MODES". Stroke the shifted +/- key, scroll down once, and stroke the key under SIZE. This brings up a prompt for a four-digit input. Input 0050.

storing coordinates

1. Input the point number and stroke

XEQ PIN

prompt: **N =?**

2. Input the north coordinate and stroke

R/S

prompt: **E =?**

3. Input the east coordinate and stroke

R/S

The coordinate has been stored under its point number.

recalling coordinates

1. Input the point number and stroke

POUT

The north coordinate is in the Y-register and the east coordinate is in the X-register. Later in this book we will be showing you a routine, "**LOAD**", which calls up "**PIN**" continuously for input of a group of coordinates.

A few more short subroutines to type in, and then we can start on the TRAVERSE program. Go to **C↑** and scroll upwards once (to 00) to begin input of these programs:

```
01▶LBL "PN?"
02 CF 22
03 "  PT NO?"
04 PROMPT
05 FS? 22
06 XEQ 01
07 RTN
08▶LBL 01
09 1
10 -
11 STO 13
12 R↓
13 RTN
14▶LBL "CODE"
15 STO 22
16 XEQ "POUT"
17 STO 24
18 R↓
19 STO 23
20 RCL 22
21 FP
*22 1E3
23 x
24 XEQ "POUT"
25 RCL- 24
26 X<>Y
27 RCL- 23
28 →POL
29 R↓
30 →HMS
31 RTN
32▶LBL "PN"
33 CLA
34 ⊢"␤     N =?"
35 PROMPT
36 CLA
37 ⊢"␤     E =?"
38 PROMPT
39 END
```

This next program isn't a short one, but the odds are, it's the one you've been anxious to get to. Stroke **GTO..** to start with a clean slate, and you can start typing it in.

a traverse program

```
00 { 859-Byte Prgm }
01▶LBL "TRAV"
02 FS?C 82
03 GTO A
04 ADV
05 XEQ "CL"
06 XEQ "F..."
07 SF 00
08 SF 01
09 SF 02
10 SF 09
11 SF 21
12 FS? 55
13 SF 08
14 CF 81
15 CLA
16 CLMENU
17▶LBL 32
18 STO 13
19 "EXIST"
20 KEY 1 GTO 01
21 "NEW"
22 KEY 2 GTO 02
23 "RE #"
24 KEY 3 GTO 03
25 MENU
26 FS? 04
27 RTN
28 "PT NO?"
29 PROMPT
30▶LBL 01
31 STO 13
32 GTO 00
33▶LBL 02
34 XEQ "PIN"
35 GTO 00
```

*To input, stroke **E** **3**

```
36▶LBL 03
37 XEQ "POUT"
38 FIX 00
39 CF 29
40 CLA
41 "NEW PT NO?"
42 PROMPT
43 STO 13
44 R↓
45 XEQ "IN"
46▶LBL 00
47 CLMENU
48 XEQ "OUT"
49 XEQ "C↑"
50 STO 08
51 R↓
52 STO 07
53▶LBL 15
54 CLMENU
55 "BRNG"
56 KEY 1 GTO 18
57 "AZIM"
58 KEY 2 GTO 19
59 "CODE"
60 KEY 3 GTO 27
61 "INV"
62 KEY 4 GTO 17
63 "INT-X"
64 KEY 5 GTO 28
65 "CLOSE"
66 KEY 6 GTO 24
67 KEY 7 GTO 31
68 KEY 8 GTO 31
69 MENU
70 RTN
71▶LBL 31
72 CLMENU
73 "H.DIS"
74 KEY 1 GTO 22
75 "S.DIS"
76 KEY 2 GTO 23
77 "SIDE"
78 KEY 3 GTO 26
79 "CURVE"
80 KEY 4 GTO 25
81 "∡ RT."
82 KEY 5 GTO 20
83 "DEF∡"
84 KEY 6 GTO 21
85 KEY 7 GTO 15
86 KEY 8 GTO 15
87 MENU
88 RTN
89▶LBL 29
90 SF 01
91 CF 02
92 GTO 31
93▶LBL 26
94 CF 01
95 SF 81
96 GTO 31
97▶LBL 27
98 SF 82
99 XEQ "CODE"
100 GTO 19
101▶LBL 28
102 SF 83
103 GTO "INT-X"
104 RTN
105▶LBL 20
106 180
107 HMS+
108 FS?C 05
109 GTO 19
110▶LBL 21
111 RCL 00
112 →HMS
113 HMS+
114 GTO 19
115▶LBL 18
116 FS? 09
117 SF 00
118 FS?C 09
119 RTN
120 SF 10
121 XEQ "A1"
122 →HMS
123▶LBL 19
124 FS? 09
125 CF 00
126 FS?C 09
127 RTN
128 →HR
129 1
130 →REC
131▶LBL 08
132 CF 10
133 →POL
134 STO 01
135 X<>Y
136 X>0?
137 GTO 07
138 360
139 +
140▶LBL 07
141 FS?C 02
142 STO 00
143 FS? 01
144 STO 00
145 STO 10
146 FC? 00
147 GTO 09
```

```
XEQ 31
XEQ "B1"
AVIEW
RTN
▶LBL 09
→HMS
XEQ "A0"
AVIEW
XEQ 31
RTN
▶LBL 23
"∡=?"
PROMPT
→HR
X<>Y
→REC
X≤Y?
X<>Y
▶LBL 22
XEQ "PN?"
RCL 00
X<>Y
FS? 01
GTO 33
RCL 10
X<>Y
▶LBL 33
FS? 01
STO+ 06
STO 01
ADV
"HD = "
ARCL ST X
FS? 08
PRON
AVIEW
ADV
→REC
FC? 01
GTO 02
STO+ 02
ENTER
ABS
STO+ 15
R↓
X<>Y
STO+ 03
ENTER
ABS
STO+ 09
R↓
2
÷
RCL 03
-
X
STO+ 04
▶LBL 02
RCL 02
FC? 01
+
RCL 07
+
FC? 01
X<>Y
RCL 03
FC? 01
+
RCL 08
+
ISG 13
STO ST X
XEQ "C+"
CF 01
SF 01
XEQ "IN"
FS? 04
GTO 40
GTO 15
RTN
▶LBL 17
X<0?
SF 07
FC?C 07
XEQ "I>"
X<0?
XEQ 34
XEQ "POUT"
1
STO- 13
R↓
RCL- 08
RCL- 03
X<>Y
RCL- 07
RCL- 02
XEQ 08
RCL 10
RCL 01
XEQ 33
RTN
▶LBL 34
+/-
"PT NO?"
PROMPT
STO 13
R↓
XEQ "IN"
RCL 13
RTN
▶LBL 24
SF 04
ADV
```

```
"Σ HD= "
ARCL 06
AVIEW
CLA
CLMENU
"OPEN"
KEY 4 GTO 36
"A"
KEY 5 GTO 37
"B"
KEY 6 GTO 38
MENU
PROFF
SF 21
"TYPE?"
AVIEW
STOP
▶LBL 36
STOP
▶LBL 37
FS? 08
PRON
RCL 04
ABS
RCL+ 05
XEQ 35
▶LBL 38
XEQ 32
"⊦  CLOSING PT#?"
PROFF
SF 21
AVIEW
STOP
STO 18
SF 10
▶LBL 01
XEQ "POUT"
GTO 04
▶LBL 02
XEQ "PIN"
▶LBL 04
XEQ "OUT"
FS? 08
PRON
ADV
CF 01
"Closure:"
AVIEW
ADV
RCL 13
SF 07
XEQ 17
AVIEW
SF 01
ADV
▶LBL 40
RCL 06
RCL÷ 01
FIX 00
CF 29
"Prcn= 1:"
ARCL ST X
AVIEW
CLMENU
CLX
CLA
SF 29
FIX 04
STOP
▶LBL 35
ADV
FS? 08
PRON
"AREA =⊦"
ARCL ST X
⊦" sq ft"
AVIEW
"= "
43560
÷
ARCL ST X
⊦" acres"
AVIEW
RTN
▶LBL 25
SF 02
ADV
GTO "CURVE"
▶LBL A
XEQ 31
RCL 18
GTO 22
END
```

south azimuth

For our users in Hawaii, Australia and parts of Canada, the program allows for the use of south azimuth. If you want to use this option, simply set flag 89.

The various routines that clear flags do **not** clear flag 89, so to leave south azmimuth you must clear flag 89 manually.

exit

If you use the traverse program as a way of setting coordinates for various points, instead of just for actual traverses (we all do) the final flag clear routines are bypassed.

```
01▶LBL "EXIT"
02 XEQ "F…"
03 81.088
04 XEQ "…F"
05 CLST
06 CLA
07 END
```

To other programs this will act as though you are still interfaced to the "**TRAV**" program.

We suggest adding the short program to the left, and then running it automatically by assigning it to key 09 in all of your 'TRAV' menus.

Add the step KEY 9 GTO "EXIT" to each of your menus, just above the "MENU" command (between 271 & 272, 86 & 87 and 68 & 69). This will clear the flags.

getting started

When you stroke XEQ TRAV the program first runs through its clearing routine and then prompts for "**PT NO?**", at the same time displaying the options menu EXIST NEW RE # .

Input the point number you want to use, and then stroke the key corresponding to the option:

1. If the point number is that of an already stored pair of coordinates stroke EXIST

2. If the coordinates have not yet been stored, stroke NEW

N =?

2a. Input the north coordinate and stroke R/S

E =?

2b. Input the east coordinate and stroke R/S

3. If this is an existing pair of coordinates , but you want to duplicate them under a new point number for this traverse, stroke **NEW**

NEW PT NO?

3a. Input the new point number you want to assign, and stroke **R/S**

The **output** will be the point number and coordinates.

#1 N= 1000.0000
E= 5000.0000

Stroking **R/S** (or any key)*brings up the traversing menu **BRNG AZIM CODE INV INT-X CLOSE** .

Output for the direction of the courses may be either BEARING or AZIMUTH. The **FIRST** (and only the first) time the traversing menu is displayed stroke either **BRNG** or **AZIM**. Nothing will seem to have happened, but your output mode is now set.

bearing input

Bearings are input in two steps, the bearing angle is entered, then the quadrant code is input.

1. Input the bearing, stroke **ENTER**

2. Input the quadrant code, and stroke **BRNG**

azimuth input

Courses may be entered as azimuths instead of bearings (if the bearing is northeast, it saves keystrokes).

1. Input the azimuth and stroke **AZIM**

input by point code

The bearing between any two **stored** coordinate pairs may be called up automatically by using "**CODE**". This 'code' consists of the two point numbers, entered in the form **AAA.bbb**, where AAA is the first point number, and bbb

*If the calculator is set to "printer on".

is the second. The second point number **must** be three-digit input.

1. Input the code for the two points, in the form noted above, and stroke 

With this method of input the 'whole' bearing is used, rather than a rounded off re-input number.

setting the direction

This initial direction may be the azimuth or bearing toward the point you want to set. That is, it may be the first direction of the traverse.

It may also be a BASIS OF BEARINGS from which an angle or deflection angle is then turned, to set the direction of the first course of the traverse. If the latter, the direction of the basis of bearings should be the direction **toward** the first point.

direction by inverse

You can also establish the direction of the first course by inversing to the next point.

1. Input the point number of the coordinates you wish to inverse to and stroke 

You can begin a traverse from existing points (in storage) by inversing from the backsight to the instrument point, to begin the traverse.

The direction of the course and the menu will be displayed, and you can choose your next move.

distance input

Distances may be input as horizontal distance or slope distance, using the appropriate keystroke.

1. Input the horizontal distance, stroke 

or

2. Input the slope distance, stroke 

∡=?

2a. Input the vertical or zenith angle, and then stroke [R/S]

The horizontal distance will be output. Stroke [R/S] if you are not using a printer, and the next prompt is displayed:

PT NO?

Except for inversing, you will have this prompt just prior to the output of coordinates, each time. You may assign any point number you want, or just stroke [R/S] to get the next consecutive point number.

CAUTION!! If you have used "**CODE**" for a direction, you must input a point number at the **PT NO?** prompt, or the point number used will be one higher than the bbb portion of the code, by default.

angle input

Angles may be turned as field angles or as deflection angles. For an angle left, stroke the [+/-] key before stroking the  or  key.

sideshots

The procedure for a sideshot is the same as for any other course, with the exception that you tell the calculator that you are wanting a sideshot.

1. <u>Signal for a sideshot</u> by stroking 

2. Input a deflection angle and stroke 

or

3. Input a field angle and stroke 

or

4a. Input a bearing, stroke [ENTER]

4b. Input the quadrant code, stroke [BRNG]

or

5. Input an azimuth, stroke [AZM]

6. Input a horizontal distance, stroke [HDS]

or

7a. Input a slope distance, stroke [SDS]

∡ =?

7b. Input the vertical or zenith angle and then stroke [R/S]

PT NO? . . . .

curved sides

It is easy to include a curved side in the traverse. The Δ angle is input as positive if the curve is to the right, and negative if the curve is to the left.

The sign of the **radius** will determine whether or not the area of the curve segment is included in the traverse area. The radius is input as a POSITIVE number to INCLUDE the area, and as a NEGATIVE number to EXCLUDE the area.

Assume that you are at the beginning of the curve, about to do the curve portion. First, establish the direction of the chord

1. Input Δ/2 and deflect (change sign if the curve is to the left) stroke [DFL]

or

2a. Input the bearing of the chord, and stroke [ENTER]

2b. Input the quadrant code, stroke [BRNG]

or

3. Input the azimuth for the curve's chord, and then stroke CHD

THEN

4. <u>Signal that you are traversing a curve by stroking</u> CURVE

DELTA R L T C D

5. Input the two known parts of the curve, stroking the appropriate keys. If the curve is to the left, Δ must be one of the two known parts in order to signal the direction of the curve. The default is a curve to the right.

   The radius must be one of the known parts if the area is to be excluded, so that it may be entered as a negative number. Default radius is positive, and the segment area is automatically included.

PT NO?

6. The next point set will be the E.C. (end of curve) coordinates*. Input the point number you wish assigned if you do not want the next consecutive number, and stroke R/S

a note on inversing

If you do sideshots and then **inverse** to the next non-sideshot point, the little lable, Inv x-y, will show the point number of the last sideshot in the 'x' position.

If you are printing out the results and want it to show the correct number, manually store the correct number into register 13 before beginning the inverse.

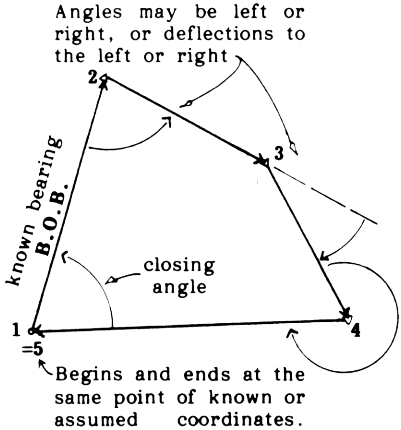
As an example, if you are at point 3, set sideshots 15, 16 and 17, and then want to inverse directly to point 4, stroke 3 STO 1 3 before 4 INV .

*A subroutine, "**RP**", is included in the "OPTIONS" section. If you add this routine to the program, the radius point will also be set.

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

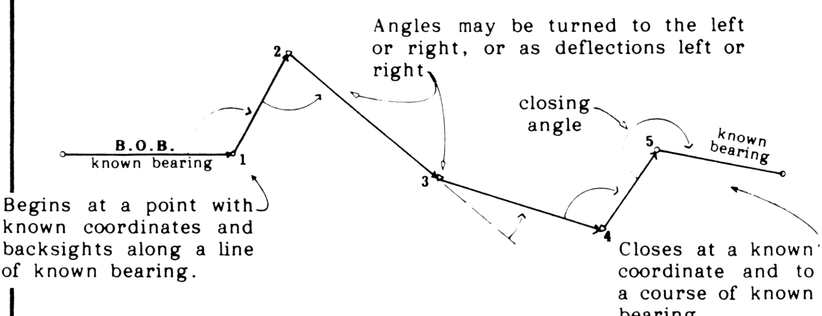
In this type of traverse, the line from **1** to **2** is usually a known line which is included in the traverse.

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.

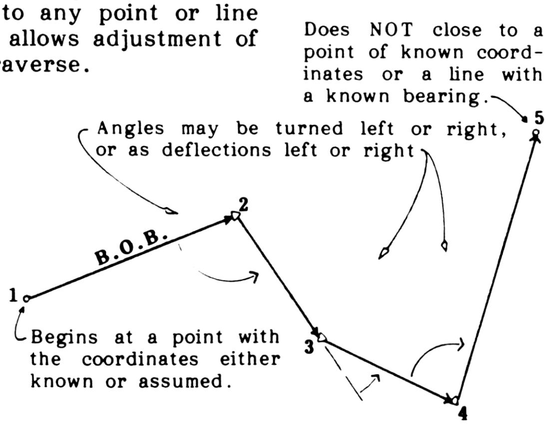


This type of traverse also closes back to the original point of beginning, and allows the turning of a closing angle, which is turned at the first (and last) point, foresighting the second point.

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 (below) 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.

More often, a traverse of this type is run as part of a topographic survey, where the traverse is considered accurate enough without correction.

For the CLOSED type of traverse, the angular error is usually distributed equally among the angles, prior to adjustment of the traverse.

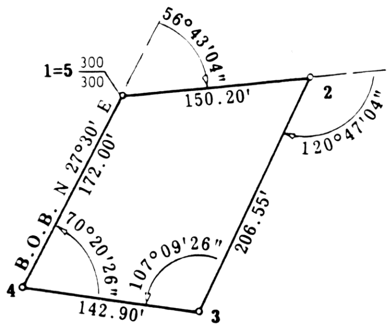
An optional program for automatic angular adjustment has been included in this book, and may be added to the traverse program later, if you will be wanting that capability in your traversing routine.

Adjustment routines for either COMPASS adjustment or TRANSIT adjustment have also been included in this book, and you can include either or both of them in your programming. You would usually want to adjust the angles of the traverse prior to using either of the adjustment routines.

We can try a keystroke example using the little traverse shown to the right.

The basis of bearings will be the course **4-1**, and we assume that these are (found) existing points which can be occupied.

The first two angles will be turned as deflection angles to the right, the last two as angles left.



Note that this is an example using a 'basis of bearings'. The course with the known bearing could also have been used directly, that is, we could have started from #4 and gone to #1 as the **first** leg of the traverse.

Stroke **XEQ** and the key that corresponds to **TRAV**

prompt: **PT NO?**

keystrokes:

[1] [NEW]

prompt: **N =?**

keystrokes:

[3] [0] [0] [R/S]

prompt: **E =?**

keystrokes:

[3] [0] [0] [R/S]

output:

```
#1   N= 300.0000
     E= 300.0000
```

keystrokes:

[BRNG] [2] [7] [.] [3] [ENTER]

[1] [BRNG]

output:

```
N 27°30'00.0" E
```

keystrokes:

[5] [6] [.] [4] [3] [0] [4]

[DEF]

output:

```
N 84°13'04.0" E
```

keystrokes:

[1] [5] [0] [.] [2] [HDIS]

prompt: **PT NO?**

keystroke:

[R/S]

output:

```
HD = 150.2000
#2   N= 315.1323
     E= 449.4358
```

keystrokes:

[1] [2] [0] [.] [4] [7] [0] [4]

[▲] [DEF]

output:

S 25°00'08.0" W

keystrokes:

[2] [0] [6] [.] [5] [5] [H.DIS]

prompt: **PT NO?**

keystroke:

[R/S]

output:

HD = 206.5500

#3 N= 127.9378
E= 362.1367

keystrokes:

[1] [0] [7] [.] [0] [9] [2] [6]

[▲] [+/-] [illegible]

output:

N 82°09'18.0" W

keystrokes:

[1] [4] [2] [.] [9] [H.DIS]

prompt: **PT NO?**

keystroke:

[R/S]

output:

HD = 142.9000

#4 N= 147.4427
E= 220.5741

keystrokes:

[7] [0] [.] [2] [0] [2] [6]

[▲] [+/-] [illegible]

output:

N 27°30'16.0" E

keystrokes:

[1] [7] [2] [H.DIS]

prompt: **PT NO?**

keystroke:

[R/S]

output:

HD = 172.0000

#5 N= 300.0024
E= 300.0067

keystroke:

[R/S] [▲] [CLOSE]

output:

Σ HD= 671.6500

prompt: **TYPE?**

This is a closed, type A, traverse so stroke the key which corresponds to "A"

output:

AREA =
24898.6429 sq ft
= 0.5716 acres

prompt: **CLOSING PT#?**

keystrokes:

[1] [illegible]

output:

Closure:

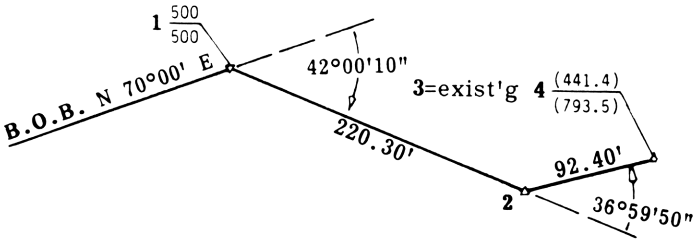
S 70°00'06.6" W

HD = 0.0072

#1 N= 300.0000
E= 300.0000

Prcn= 1:93818

As another example of starting with a 'basis of bearings', this type "B" traverse starts by backsighting down a known course from a point with known coordinates (#1, N 500/E 500). In the example, traverse point #3 closes at a point with known coordinates (#4, N 441.4/E 793.5).



Begin by stroking **XEQ**, and then the key which corresponds to "**TRAV**".

prompt: **PT NO?**

keystrokes:

1 [NEW]

prompt: **N =?**

keystrokes:

5 0 0 R/S

prompt: **E =?**

keystrokes:

5 0 0 R/S

output:

```
#1   N= 500.0000
     E= 500.0000
```

keystrokes:

[BRNG] 7 0 ENTER

1 [BRNG]

output:

```
N 70°00'00.0" E
```

keystrokes:

4 2 . 0 0 1 [DEFL]

output:

```
S 67°59'50.0" E
```

keystrokes:

2 2 0 . 3 [HDIS]

prompt: **PT NO?**

keystroke:

R/S

output:

```
HD = 220.3000

#2   N= 417.4643
     E= 704.2546
```

keystrokes:

3 6 . 5 9 5 +/-

▼ [DEFL]

output:

```
N 75°00'20.0" E
```

keystrokes:

9 2 . 4 [HDIS]

prompt: **PT NO?**

keystroke:

R/S

output:

HD = 92.4000

#3 N= 441.3705
E= 793.5085

keystrokes:

R/S ▲ [illegible]

output:

Σ HD= 312.7000

prompt: **TYPE?**

This is a closed traverse, type B, so stroke the key which corresponds to "B"

prompt: **CLOSING PT#?**

keystrokes:

4 [illegible]

prompt: **N =?**

keystrokes:

4 4 1 . 4 R/S

prompt: **E =?**

keystrokes:

7 9 3 . 5 R/S

output:

Closure:

N 16°00'32.3" W

HD = 0.0307

#4 N= 441.4000
E= 793.5000

Prcn= 1:10186

Because this was a type "B" traverse, no area is output, but a closure **is** calculated. If this had been an "open" traverse, no closure would be given since it does not close to a known point.

NOTE: If the function you want to use is not in the displayed menu when you need it, you can use ▼ or ▲ to go to the other menu. If you are not using a printer while running the program, stroke R/S for the next output or prompt.

miscellaneous moves

You can also inverse to **unstored** coordinates by entering the northing, then input the easting and stroke the +/- key before you stroke INV. The program will prompt for a point number to be assigned, and then store the coordinates.

Once the traverse has been closed it may be adjusted. We have included two programs for traverse adjustment in this book, "**COMP**ass adjustment" and "**TRAN**sit adjustment".

You will probably only want to put in one of these, and which one you use will depend on the equipment you use for traversing. With a good theodolite and EDM (or a total station) Compass Adjustment is most often used.

subroutines

First, two subroutines which are used by either of the adjustment programs. Go to "**C↑**", scroll up to 00 and type in the programs shown to the right.

"**INV**" performs an inversing routine, and is used by an intersection solutions program and a coordinate transformation program later in this book.

The program "**BLK**" is also used by the transformation program as well as several others. It identifies a block of consecutive points by point number for coordinates manipulation.

Because these are not routines that you will execute from the keyboard, we have placed them at a point in program memory which keeps them out of the way in the program menu.

```
▶LBL "INV"
RCL- 08
X<>Y
RCL- 07
→POL
STO 23
X<>Y
360
X<>Y
X<0?
+
FC? 00
XEQ 01
FS? 00
XEQ "B1"
├"␊HD = "
ARCL 23
AVIEW
ADV
RTN
▶LBL 01
→HMS
1
XEQ "A1"
RTN
▶LBL "BLK"
"Begin Pt?"
├"␊"
PROMPT
"Last Pt#?"
├"␊"
PROMPT
1
+
STO 22
X<>Y
END
```

As always, it is important that you proof-read what you have typed in, before exiting the program.

Stroke the shifted **XEQ** key and then the key which corresponds to "**TRAV**". Enter program mode by stroking the shifted **R/S** key, scroll upward to program step 00 and type in either "**COMP**" or "**TRAN**", from the program listings.

COMPass adjustment

```
01▸LBL "COMP"
02 ADV
03 "COMPASS ADJUST"
04 FS? 08
05 AVIEW
06 ADV
07 0
08 STO 11
09 STO 14
10 STO 15
11 XEQ "BLK"
12 XEQ "POUT"
13 XEQ "C↑"
14 ADV
15 1
16 STO+ 13
17 RCL 10
18 RCL 01
19 +/-
20 →REC
21 RCL÷ 06
22 +/-
23 STO 16
24 X<>Y
25 RCL÷ 06
26 +/-
27 STO 05
28▸LBL 02
29 RCL 13
30 RCL 22
31 X=Y?
32 GTO 01
33 XEQ "OUT"
34 STO 01
35 STO 15
36 X<>Y
37 STO 00
38 STO 14
39 X<>Y
40 RCL- 08
41 X<>Y
42 RCL- 07
43 →POL
44 STO+ 11
45 RCL 11
46 RCL× 16
47 STO+ 14
48 RCL 11
49 RCL× 05
50 STO+ 15
51 RCL 14
52 RCL 15
53 XEQ "IN"
54 RCL 13
55 1
56 -
57 XEQ "POUT"
58 STO 08
59 X<>Y
60 STO 07
61 1
62 STO+ 13
63 XEQ "OUT"
64 XEQ "INV"
65 RCL 00
66 STO 07
67 RCL 01
68 STO 08
69 XEQ "OUT"
70 XEQ "C↑"
71 ISG 13
72 STO ST X
73 ADV
74 GTO 02
75▸LBL 01
76 STOP
77 END
```

Typing in the last step, END, will automatically separate the program from "**TRAV**", but still leaves it just to the right of the "TRAV" key in the menu.

TRANsit adjustment

```
01▶LBL "TRAN"
02 ADV
03 "TRANSIT ADJUST"
04 FS? 08
05 AVIEW
06 ADV
07 SF 05
08 SF 06
09 RCL 02
10 X<0?
11 CF 05
12 RCL 03
13 X<0?
14 CF 06
15 0
16 STO 11
17 STO 14
18 XEQ "BLK"
19 XEQ "POUT"
20 XEQ "C↑"
21 ADV
22▶LBL 03
23 1
24 STO+ 13
25 RCL 13
26 RCL 22
27 X=Y?
28 GTO 01
29 XEQ "OUT"
30 ENTER
31 RCL- 08
32 RCL× 03
33 RCL÷ 09
34 ABS
35 FS? 06
36 +/-
37 STO+ 14
38 CLX
39 RCL 14
40 +
41 STO 19
42 R↓
43 ENTER
44 RCL- 07
45 RCL× 02
46 RCL÷ 15
47 ABS
48 FS? 05
49 +/-
50 STO+ 11
51 CLX
52 RCL 11
53 +
54 RCL 19
55 XEQ "IN"
56 RCL 13
57 1
58 -
59 STO 13
60 XEQ "OUT"
61 STO 08
62 X<>Y
63 STO 07
64 1
65 STO+ 13
66 XEQ "OUT"
67 STO 01
68 X<>Y
69 STO 00
70 X<>Y
71 ADV
72 XEQ "INV"
73 RCL 00
74 STO 07
75 RCL 01
76 STO 08
77 XEQ "OUT"
78 XEQ "C↑"
79 GTO 03
80▶LBL 01
81 STOP
```

For program step #82, type in **XEQ END**.

If you have decided to input BOTH types of adjustment, you should put in "**TRANS**" above "**COMP**". You can eliminate program steps 80 and 81 from "**TRANS**" and substitute **RTN** for **END** as the last step. This lets the two routines share LBL 01 instead of duplicating the steps.

```
 #1  N= 300.0000
     E= 300.0000

N 27°30'00.0" E
N 84°13'04.0" E

HD = 150.2000

 #2  N= 315.1323
     E= 449.4358

S 25°00'08.0" W

HD = 206.5500

 #3  N= 127.9378
     E= 362.1367

N 82°09'18.0" W

HD = 142.9000

 #4  N= 147.4427
     E= 220.5741

N 27°30'16.0" E

HD = 172.0000

 #5  N= 300.0024
     E= 300.0067
```

As an example, we have re-run the traverse on page 29, and the output from both types of correction is shown below.

After you have closed the traverse, stroke **XEQ** and then the key corresponding to either "**COMP**" or "**TRAN**".

prompt: **BEGIN @?**

1. Input the beginning point number, then stroke R/S

prompt: **LAST PT#?**

2. Input the last point number in the traverse, stroke R/S

TRANSIT ADJUST	COMPASS ADJUST
#1 N= 300.0000 E= 300.0000	#1 N= 300.0000 E= 300.0000
N 84°13'03.8" E HD= 150.1978	N 84°13'04.5" E HD= 150.1984
#2 N= 315.1322 E= 449.4336	#2 N= 315.1317 E= 449.4343
S 25°00'08.6" W HD= 206.5517	S 25°00'09.6" W HD= 206.5516
#3 N= 127.9365 E= 362.1332	#3 N= 127.9365 E= 362.1331
N 82°09'18.6" W HD= 142.9020	N 82°09'19.0" W HD= 142.9013
#4 N= 147.4413 E= 220.5686	#4 N= 147.4409 E= 220.5691
N 27°30'15.3" E HD= 171.9986	N 27°30'14.5" E HD= 171.9986
#5 N= 300.0000 E= 300.0000	#5 N= 300.0000 E= 300.0000

The solutions to intersection 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 PI or the new lot corner. Or, we need to know how far a point is offset from a given line.

Next to just plain traversing, this is the most used type of calculation in surveying. We've tried to make it easy, with all of the options displayed in the menu at one time.

Any distance input is done with the distance key, and any direction can be input as bearing, azimuth or code.

The "CODE" key may be used to recall a "stored" bearing between two points in storage. The point numbers are input in the form **AAA.bbb**, where **AAA** is the first point number and **bbb** is the second. Three digit input is required for the second point number.

to use the program

Begin by stroking **XEQ**, and then the key corresponding to **INT-X**

Begin @ PT#?

1. Input the beginning point number, stroke R/S

OUTPUT will be the point number and coordinates of the beginning point.

End Pt#?

2. Input the point number of the ending point, then stroke R/S

Save as #?

3. Input the point number you wish to assign to the intersection point, then stroke R/S

The prompts and responses on the previous page are the same for all of the intersection routines. Select the type of intersection you need, and follow the keystroke instructions below.

bearing - bearing

1a. Input the first bearing and stroke **ENTER**

Input the quadrant code, stroke 

or

1b. Input the point code for the bearing you want to extract as the first bearing, and then stroke 

or

1c. Input the azimuth of the first course and stroke 

2a. Input the bearing of the second line, then stroke **ENTER**

Input the quadrant code and stroke 

or

2b. Input the point code for the bearing you want to extract as the second bearing, and stroke 

or

2c. Input the azimuth of the second course and stroke 

OUTPUT will be the bearing and distance from the beginning point to the intersection, the point number and coordinates of the intersection point, the bearing and distance from the intersection point to the end point, then the point number and coordinates of the last point.

bearing - distance

1a. Input the bearing of the first course, stroking

ENTER

Input the quadrant code, then stroke

BRNG

or

1b. Input the point code for the bearing you want to extract, and stroke

PNT

or

1c. Input the azimuth of the first course, and then stroke

AZM

2. Input the distance for the second line, and stroke

DST

OUTPUT will be the bearing and distance from the beginning point to the intersection, the point number and coordinates of the intersection point, the bearing and distance from the intersection point to the end point, then the point number and coordinates of the last point.

Because there are two possible answers with this solution type, a reminder prompt appears

2nd Solution

YES NO

3. Examine the answers and decide if they are the correct solution. If they are not, go on to the second solution by stroking

YES

Output will be the same as for the first solution with the exception that the first point is not printed out again. The intersection point coordinates in storage will be replaced by the new ones.

distance - distance

1. Input the first distance, stroke **DIS**

2. Input the second distance, and then stroke **DIS**

OUTPUT will be the bearing and distance from the beginning point to the intersection, the point number and coordinates of the intersection point, the bearing and distance from the intersection point to the end point, then the point number and coordinates of the last point.

2nd Solution

YES **NO**

3. If you want the second solution, stroke **YES**

   If you do not want the second solution, stroke **NO**

Output of the second solution is similar to that of the bearing - distance solution, replacing the original coordinates at the intersection point with the new ones.

offset to a line

1a. Input the bearing of the known line (from which the end point is offset) and stroke **ENTER**

   Input the quadrant code then stroke **QUAD**

or

1b. Input the code for the bearing you want to extract and stroke **CODE**

or

1c. Input the azimuth of the line and stroke **AZIM**

2. Stroke the key which corresponds to

OUTPUT of the answer will be in the same form as the other intersection routines, with the bearings and distances of both lines given.

the programming

The information on the last few pages describes the way the program works, and now we'll look at the programming itself.

As always, there are a couple of related subroutines.

2 ENTER 1

MATRIX NEW EXIT

STO ENTER ENTER

This time you will also add a few steps to the traverse program and set up a storage matrix. That is much easier than it sounds, just do the keystrokes shown to the left.

Go to "**INV**", enter program mode, and scroll up one notch to step 00 and type in the following:

```
01▶LBL "B→A"        08 RCL 21
02 XEQ "A1"         09 ⊢"↓"
03 ⊢"↓"             10▶LBL 01
04 GTO 01           11 FS? 38
05▶LBL "A→B"        12 RTN
06 →HR              13 AVIEW
07 XEQ "B1"         14 RTN
```

Next, stroke

XEQ TRAV

Enter program mode.

Scroll down until the program pointer is at 04, and insert the two steps shown to the right.

```
05 FS?C 83
06 GTO 15
```

And, it's time to type in another long one. Before starting stroke [ ] XEQ • • to pack what is already in the calculator. Now go to the permanent .**END**. by stroking shift, XEQ, .END., and enter program mode. The display should show

```
00 { 0-Byte Prgm }
01▸.END.
```

Stroke ▲ so that the pointer is at 00, and you may begin typing in the program steps listed below.

```
01▸LBL "INT-X"
02 XEQ d
03 FS? 83
04 XEQ 51
05 SF 21
06 CLA
07 XEQ 10
08 CLMENU
09▸LBL 50
10 "BRNG"
11 KEY 1 GTO F
12 "CODE"
13 KEY 2 GTO A
14 "AZIM"
15 KEY 3 GTO B
16 "DIST"
17 KEY 4 GTO G
18 "O/S"
19 KEY 5 GTO D
20 "TRAV"
21 KEY 6 GTO C
22 MENU
23 STOP
24▸LBL d
25 86.099
26▸LBL H
27 CF IND ST X
28 ISG ST X
29 GTO H
30 CLX
31 RTN
32▸LBL 51
33 INDEX "SW"
34 RCL 07
35 STOEL
36 ↓
37 RCL 08
38 STOEL
39 CLX
40 RTN
41▸LBL C
42 INDEX "SW"
43 RCLEL
44 STO 07
45 STO- 02
46 ↓
47 RCLEL
48 STO 08
49 STO- 03
50 RCL 19
51 →HR
52 180
53 +
54 STO 10
55 STO 00
56 CLX
57 GTO "TRAV"
58▸LBL A
59 XEQ "CODE"
60▸LBL B
61 1
62▸LBL F
63 SF 10
64 XEQ "A1"
65▸LBL e
66 FC? 95
```

```
67 STO 04
68 FS? 95
69 STO 00
70 CLX
71 FS? 95
72 SF 97
73 SF 95
74 FC? 97
75 RTN
76 FS? 97
77 GTO a
78 RTN
79▶LBL G
80 FC? 98
81 STO 05
82 FS? 98
83 STO 07
84 FS? 95
85 STO 07
86 FS? 95
87 GTO b
88 FS? 98
89 SF 99
90 FS? 99
91 GTO c
92 SF 98
93 CLX
94 RTN
95▶LBL D
96 SF 96
97 XEQ 00
98 RCL 04
99 RCL- 10
100 SIN
101 RCLx 11
102 STO 07
103▶LBL b
104 CF 95
105 XEQ 00
106 RCL 07
107 X↑2
108 RCL 04
109 RCL- 10
110 RCL 11
111 →REC
112 FS? 96
113 GTO 05
114 R↓
115 X↑2
116 -
117 SQRT
118 R↑
119 X<>Y
120 -
121 STO 12
122 LASTX
123 R↑
124 +
125▶LBL 05
126 STO 05
127 FS?C 96
128 GTO 02
129 XEQ 02
130 RCL 05
131 X<> 12
132 STO 05
133 XEQ 01
134 GTO 02
135▶LBL 10
136 CLA
137 "Begin @ Pt#?"
138 ⊢"↳"
139 FC? 83
140 PROMPT
141 FC? 83
142 XEQ "POUT"
143 FS? 83
144 XEQ "OUT"
145 STO 01
146 R↓
147 STO 06
148 CF 10
149 R↑
150 FC? 83
151 XEQ "C↑"
152 "End Pt#?"
153 ⊢"↳"
154 PROMPT
155 STO 16
156 XEQ "POUT"
```

```
157 STO 03
158 R↓
159 STO 02
160 "Save as #?"
161 ⊢"↳"
162 PROMPT
163 STO 15
164 CF 95
165 SF 10
166 FIX 04
167 CLX
168 RTN
169▶LBL 00
170 RCL 03
171 RCL- 01
172 RCL 02
173 RCL- 06
174 XEQ 03
175 STO 10
176 X<>Y
177 STO 11
178 RTN
179▶LBL a
180 CF 95
181 XEQ 00
182 RCL 00
183 RCL- 10
184 SIN
185 RCL× 11
186 RCL 00
187 RCL- 04
188 SIN
189 ÷
190 STO 05
191 GTO 02
192▶LBL c
193 CF 95
194 XEQ 00
195 RCL 05
196 →POL
197 X↑2
198 RCL 07
199 X↑2
200 -
201 RCL÷ 11
202 RCL÷ 05
203 2
204 ÷
205 ACOS
206 RCL 10
207 X<>Y
208 -
209 STO 12
210 LASTX
211 RCL+ 10
212 STO 04
213 XEQ 02
214 RCL 12
215 X<> 04
216 STO 12
217 XEQ 01
218▶LBL 02
219 CF 10
220 RCL 04
221 RCL 05
222 →REC
223 XEQ 03
224 ADV
225 RCL 04
226 RCL 05
227 →REC
228 RCL+ 06
229 STO 08
230 X<>Y
231 RCL+ 01
232 STO 09
233 RCL 15
234 STO 13
235 R↓
236 XEQ "IN"
237 XEQ "OUT"
238 XEQ "C↑"
239 RCL 02
240 RCL 03
241 RCL- 09
242 X<>Y
243 RCL- 08
244 XEQ 03
245 RCL 16
246 STO 13
```

```
247 RCL 02
248 RCL 03
249 ADV
250 XEQ "C↑"
251 RTN
252 GTO 50
253▶LBL 03
254 →POL
255 STO 14
256 X<>Y
257 XEQ 04
258 FS? 10
259 RTN
260 ADV
261 →HMS
262 SF 88
263 XEQ "A→B"
264 CF 88
265 FIX 03
266 ⊢"Dist = "
267 ARCL 14
268 AVIEW
269 FIX 04
270 RTN
271▶LBL 01
272 CLMENU
273 "2nd Solution"
274 CF 21
275 AVIEW
276 XEQ "YN"
277 SF 21
278 FS?C 10
279 RTN
280 STOP
281▶LBL 04
282 X>0?
283 RTN
284 360
285 +
286 RTN
287 .END.
```

As always, it's important to proof-read the program before trying it out.

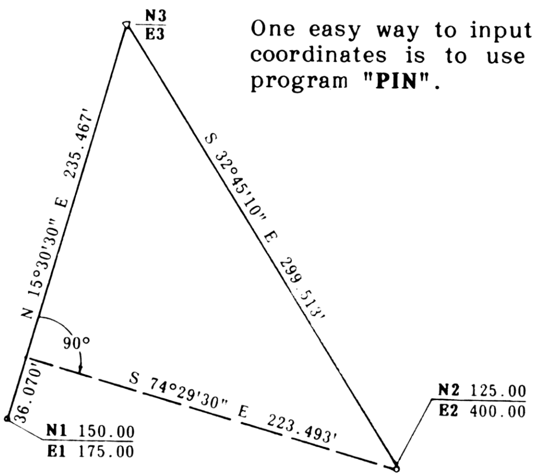
Then try the keystroke examples on the following pages, to get the feel of the different routines.

NOTE: Distance-Distance and Bearing-Distance solutions can "blow" if the angle of intersection is close to 90°! From 89°58'55" to 90°00'00" the sine is 1 to the nearest 7 places, and the cosine is 0 at 90°, not allowing division, since the calculator is programmed to not divide if the number in the x-register is 0.

Quite often, when this happens, the correct answer lies half-way between the answers output as the 1st and 2nd solutions. If both solutions lie in the same quadrant and the coordinates are <u>very</u> close in both results, try using the mean northing and easting by averaging the coordinates.

Before beginning with the keystroke examples, we need to store points **1** and **2** in the illustration below.

One easy way to input the coordinates is to use the program "**PIN**".



We'll start with an example using the **bearing-bearing** routine.

prompt: **Begin @ Pt#?**

keystrokes:

1 R/S

output: #1 N= 150.0000
E= 175.0000

prompt: **End Pt#?**

keystrokes:

2 R/S

prompt: **Save as #?**

keystrokes:

3 R/S

display: 0.0000

BRNG CODE AZIM DIST O/S TRAV

keystrokes:

1 5 . 3 0 3 ENTER

1 BRNG

display: 0.0000

BRNG CODE AZIM DIST O/S TRAV

keystrokes:

3 2 . 4 5 1 ENTER

2 BRNG

output:

N 15°30'29.8" E
Dist = 235.467
#3 N= 376.8941
E= 237.9586

S 32°45'10.2" E
Dist = 299.513
#2 N= 125.0000
E= 400.0000

Next, try an example of the **offset** routine, using the same illustration for this example.

keystrokes:

[XEQ] [INTER]

prompt: **Begin @ Pt#?**

keystrokes:

[1] [R/S]

output:

```
#1   N= 150.0000
     E= 175.0000
```

prompt: **End Pt#?**

keystrokes:

[2] [R/S]

prompt: **Save as #?**

keystrokes:

[3] [R/S]

display: 0.0000

[BRNG] [CODE] [AZIM] [DIST] [O/S] [TRAV]

Since the bearing that we want (from 1 to 3) is still stored from the last example, let's just call it up from storage:

keystrokes:

[1] [.] [0] [0] [3] [CODE]

When the display clears to 0.0000 and the prompt bar appears, indicate that this is an OFFSET:

keystroke:

[O/S]

output:

```
N 15°30'30.0" E
Dist = 36.070

#3   N= 184.7571
     E= 184.6444

S 74°29'30.0" E
Dist = 223.493

#2   N= 125.0000
     E= 400.0000
```

For a last example, which also uses the 'second solution', do the **distance-distance** intersection, using the same coordinates as in the illustration, and the distances which were output in the first example.

Answer the beginning prompts with the same point numbers as in the other two examples, and then input the first distance

keystrokes:

[2] [3] [5] [.] [4] [6] [7]

[DIST]

When the display clears to 0.0000 and the prompt bar appears, input the second distance

keystrokes:

[2] [9] [9] [.] [5] [1] [3]

[DIST]

The output which follows is not the correct answer for the directions we are going:

output:

```
S 2°49'40.4" E
Dist = 235.467
 #3  N= -85.1803
     E= 186.6170

N 45°25'59.6" E
Dist = 299.513
 #2  N= 125.0000
     E= 400.0000
```

prompt: **2nd Solution**

YES NO

keystroke:

YES

output:

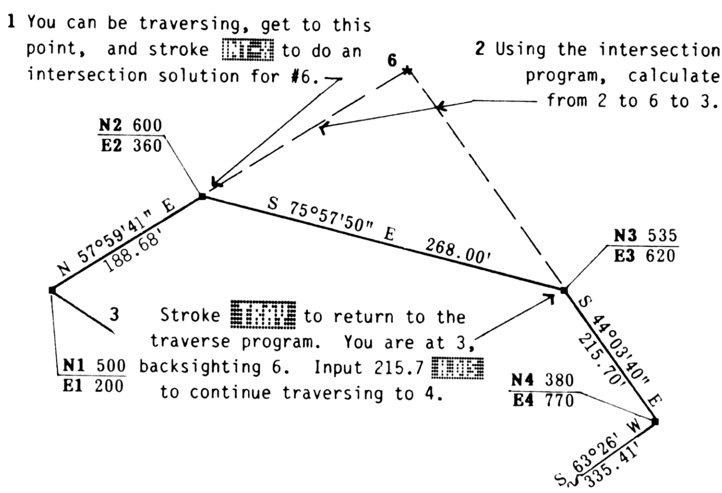
```
N 15°30'29.8" E
Dist = 235.467
 #3  N= 376.8941
     E= 237.9586

S 32°45'10.2" E
Dist = 299.513
 #2  N= 125.0000
     E= 400.0000
```

If you are not using a printer while working this routine, you **must** continue to stroke the R/S key after output, until a new prompt appears or the display clears.

interfacing with "TRAV"

You will have noticed that "**TRAV**" is also displayed in your prompt bar. You cannot go directly to the traverse program from this program unless you originally started out there.

What you CAN do is go to the intersection program **from** the traverse program, and then return to the traverse.



DECISIONS, DECISIONS, DECISIONS!!

We have arrived at a point where you have to decide how to configure your calculator.

point storage

If you have input all of the programs so far, you should be able to store about 210 points, and you have a good basic set of programs for most traverse problems.

From here on, each program you add is a trade for point storage. Decide which programs you really need and how many stored points you actually need for YOUR normal work. To help with this we will tell you how many points each of the following programs will replace.

a recommendation

Adding the routine to the right to the traverse program, and changing the two steps shown, will replace 19 registers, but allows you to use "**TRIG**", "**A→B**" and "**B→A**" without leaving the traverse program. Typing in the programs on the next page adds three valuable programs to your set.

The new routine should go directly above LBL 29 in the traverse program, the other two steps replace existing steps. If you did not do the modification for the intersection program, the step numbers will be 2 smaller.

```
92▸LBL 60
93 CLMENU
94 "SPRAY"
95 KEY 1 GTO "SPRAY"
96 "LOAD"
97 KEY 2 GTO "LOAD"
98 "DUMP"
99 KEY 3 GTO "DUMP"
100 "TRIG"
101 KEY 4 GTO 61
102 "B→A"
103 KEY 5 GTO 62
104 "A→B"
105 KEY 6 GTO 63
106 KEY 7 GTO 31
107 KEY 8 GTO 15
108 MENU
109 RTN
110▸LBL 61
111 SF 83
112 XEQ "TRIG"
113 GTO 15
114▸LBL 62
115 XEQ "B→A"
116 GTO 15
117▸LBL 63
118 XEQ "A→B"
119 GTO 15
```

Replace `70 KEY 7 GTO 31` with `70 KEY 7 GTO 60`

Replace `89 KEY 8 GTO 15` with `89 KEY 8 GTO 60`

We feel that you'll get a lot of use out of these three short programs, particularly "**SPRAY**", which is used for radial stakeout.

```
01▶LBL "DUMP"
02 XEQ "BLK"
03 X<>Y
04 1
05 -
06 1E3
07 ÷
08 +
09 STO 13
10▶LBL B
11 XEQ "OUT"
12 XEQ "C↑"
13 ISG 13
14 GTO B
15 RTN
16▶LBL "LOAD"
17 KEY 9 GTO 10
18 "Point #?↯"
19 PROMPT
20 XEQ "PIN"
21 GTO "LOAD"
22▶LBL "SPRAY"
23 KEY 9 GTO 10
24 SF 21
25 XEQ "CL"
26 CF 00
27 "Inst @?"
28 ⊢"↯"
29 PROMPT
30 XEQ "POUT"
31 STO 08
32 R↓
33 STO 07
34 "Inst @ "
35 XEQ 09
36 FS? 55
37 AVIEW
38 ADV
39▶LBL A
40 CLA
41 "Pt #?↯"
42 PROMPT
43 XEQ "POUT"
44 CLA
45 XEQ 09
46 RCL- 08
47 X<>Y
48 RCL- 07
49 →POL
50 STO 23
51 X<>Y
52 X<0?
53 XEQ 01
54 →HMS
55 ⊢"   AZ: "
56 XEQ "DMS"
57 ⊢"↯    HD: "
58 ARCL 23
59 AVIEW
60 ADV
61 GTO A
62▶LBL 01
63 360
64 +
65 RTN
66▶LBL 09
67 FIX 00
68 CF 29
69 ARCL 13
70 FIX 04
71 SF 29
72 RTN
73▶LBL 10
74 CLMENU
75 EXITALL
76 END
```

All three may be accessed directly from your traverse program, using the new menu (page 49). A short description of each follows.

dump

```
#1   N= 1000.0000
     E= 5000.0000

#2   N= 1254.0000
     E= 5100.0000

#3   N= 902.0000
     E= 4825.0000

#4   N= 865.0000
```

When you stroke DUMP, you are prompted for the first and last point numbers you wish to output.

The calculator will print out a complete list (left) of all of the coordinates within that block of stored coordinates.

The printout can be made on the infra-red printer or through your computer, if you are using one of Rush Systems' "hookups". With the hookup, save a little time by setting the "delay" to 0 on your calculator first.

load

This one runs a closed loop on "**PIN**" for input of coordinates. When you execute this program you will get a new prompt for point number immediately after input of each coordinate pair, eliminating the need for executing "**PIN**" each time when you want to input a group of coordinates.

spray

When this program is executed it first prompts for the point number of the instrument's location, then prompts for the next point number continuously, inversing to each of the points.

The output is as shown to the right ("Inst @ **X**" is only output if a printer is being used).

```
Inst @ 2

1    AZ: 201°29'22.6"
     HD: 272.9762

3    AZ: 217°59'55.4"
     HD: 446.6867

4    AZ: 214°09'47.9"
     HD: 470.1245
```

Inverse to the backsight first, setting that azimuth in the gun when you backsight. Turn the azimuth of each of the other points to lay them out.

If you want to lay out a group of unstored points from a provided "dump sheet", simply input them with "**LOAD**" and then inverse them with"**SPRAY**".

purging a file

This is another handy utility program, and it's practically free, using only 21 bytes of program space.

Use it to clear a block of stored coordinates when you need to make room for new calculations.

Input the 'code' form for the block of points you want to clear, and THEN execute "**PRGE**" (short for **PuRGE**).

The block of coordinates is cleared by storing 0 into each of the storage registers.

```
01▶LBL "PRGE"
02 STO 13
03▶LBL F
04 CLST
05 XEQ "IN"
06 ISG 13
07 GTO F
08 END
```

Before clearing the coordinates, you might want to print them out, so that you have a permanent record of them. Use "**DUMP**" for this. Incidentally, if you want more places shown in the output, you can change the output with FIX 06 or FIX 08 before executing "**DUMP**"(See pg. 12).

setting the radius point

The footnote on page 26 mentioned this program; it adds the radius point to the output when you traverse a curve. It's a trade for about 5 stored points if you add it. Because it is NOT a routine you will be calling up directly, you can add it anywhere in the middle of program memory.

Also, because it doesn't have any local labels, it isn't necessary to have it as a separate program, and it can be tacked on to any of the other subroutines.

```
insert
210 FS? 82
211 XEQ "RP"
        208 ..
   here 209 STO+ 05
        210 RCL 18
        211 CLMENU
        212 GTO "
```

Before starting to type in "**RP**", modify the program, "**CURVE**" as shown to the left by adding in two additional program steps.

The subroutine is actually called from this program while you are traversing a curve.

```
00 { 79-Byte Prgm }
01▶LBL "RP"
02 CLMENU
03 RCL 00
04 RCL 12
05 FS? 05
06 +/-
07 -
08 90
09 FS?C 05
10 +/-
11 +
12 RCL 11
13 →REC
14 RCL+ 07
15 RCL+ 02
16 X<>Y
17 RCL+ 08
18 RCL+ 03
19 "Rad. Pt#?⊦"
20 PROMPT
21 STO 13
22 R↓
23 XEQ "IN"
24 XEQ "OUT"
25 ADV
26 "Radius Pt:"
27 FS? 55
28 AVIEW
29 XEQ "C↑"
30 RTN
```

The routine is run automatically, but there are a couple of things that you must remember to do.

When the prompt **"Rad. Pt#?"** appears, you MUST input a point number before stroking the RUN button.

When the program returns to the traverse routine and prompts for a point number after output of the chord, you MUST input a point number.

```
DELTA= [illegible]
    L= 174.5329
    T= 88.1635
    C= 173.6482

Radius Pt:
 #2     N= 128.9899
        E= 769.8463

HD = 173.6482

 #3     N= 450.3837
        E= 386.8241
```

You can not just stroke the RUN button for the next consecutive point number. Output is as shown above.

additional programming

The pages which follow contain additional programs which you may add as options to what you already have in your calculator. The number of stored points traded by adding these programs is as follows:

Curve **LA**Yout . . . . ±29 points

Coordinate **ROT**ation . . . . ±39 points

PREdetermined Areas . . . . ±33 points

automatic angle adjustment

After a traverse is closed, but before adjustment by compass or transit method, the angles should be balanced.

If you will refer back to the traverse example on page 30, you'll note that the original basis of bearings was N 27°30'00" E.

After turning the angles through the traverse we ended up with the same line, but it now shows the bearing as N 27°30'16" E, indicating that we have 16" too much angle in the traverse.

When you divide the 16" by the 4 traverse points, you get an angular error of 4" per turn.

While this is an acceptable amount of angular error, it still needs to be adjusted out. That's what this program does.

```
#1  N= 300.0000
    E= 300.0000

N 27°30'00.0" E
N 84°13'04.0" E

HD = 150.2000

#2  N= 315.1323
    E= 449.4358

S 25°00'08.0" W

HD = 206.5500

#3  N= 127.9378
    E= 362.1367

N 82°09'18.0" W

HD = 142.9000

#4  N= 147.4427
    E= 220.5741

N 27°30'16.0" E

HD = 172.0000

#5  N= 300.0024
    E= 300.0067
```

The program actually recalls, then adjusts, each bearing in the traverse. More, having adjusted the bearing (or azimuth), it goes on to use the new bearing to recalculate (and store) the coordinates of each traverse point.

```
         Go to
314▸LBL 38
315 XEQ "∠ADJ"
     add this step
```

This subroutine is called up directly from the closure routine in the traverse program. One step (see left) has to be added to the traverse program.

Stroke shift, XEQ (GTO), TRAV then shift XEQ 38. Enter program mode, and the program pointer should be at "LBL 38". The step number may be different, depending on which options you've already installed, but the important thing is that the new step be the first step in LABEL 38.

Type in the new step, and we'll go on to type in the subroutine.

The program can go anywhere into the middle of the subroutine stack. Go to any of the subroutines that starts with program step 01, scroll up to 00, and begin the input.

```
01▸LBL "∡ADJ"
02 RCL 13
03 STO 24
04 1
05 +
06 STO 22
07 CF 22
08 "Error/Turn?⊦"
09 PROMPT
10 FC? 22
11 RTN
12 1E4
13 ÷
14 →HR
15 +/-
16 STO 11
17 CLX
18 STO 12
19 "Beginning Pt#?⊦"
20 PROMPT
21 STO 23
22 XEQ "POUT"
23 STO 08
24 STO 14
25 R↓
26 STO 07
27 STO 16
28▸LBL A
29 RCL 22
30 RCL 13
31 X=Y?
32 GTO B
33 XEQ "POUT"
34 RCL- 08
35 X<>Y
36 RCL- 07
37 →POL
38 X<>Y
39 RCL 11
40 RCL× 12
41 +
42 X<>Y
43 →REC
44 STO+ 16
45 X<>Y
46 STO+ 14
47 XEQ "OUT"
48 STO 08
49 R↓
50 STO 07
51 RCL 16
52 RCL 14
53 XEQ "IN"
54 1
55 STO+ 12
56 STO+ 13
57 GTO A
58▸LBL B
59 RCL 23
60 XEQ "POUT"
61 STO 08
62 R↓
63 STO 07
64 RCL 24
65 XEQ "POUT"
66 RCL- 08
67 STO 03
68 R↓
69 RCL- 07
70 STO 02
```

Now, type in XEQ END, for program step 71. This assures that you won't branch to the wrong "LABEL A" or "LABEL B" by separating this program from the others.

In order to angle adjust a type "B" traverse such as the one on pages 31 and 32, it would be necessary to occupy point #3 = #4(existing) and turn a closing angle to the known course to determine the angular error.

With this routine installed, part of the traverse closure program will prompt:

Error/Turn?

If you DO NOT want to adjust the angles, return to the closure by stroking R/S

If you DO want to adjust the angles, input the number of **seconds** of error per turn, and then stroke R/S

Beginning Pt#?

This prompt will appear if you ARE adjusting the angles. Input the point number used for the first traverse point, and stroke R/S

```
Closure:
N 31°03'29.4" W
HD = 0.0007
 #1  N= 300.0000
     E= 300.0000
Prcn= 1:914501
```

The calculator will do the adjusting of the angles **and** coordinates, then continue with the closure. For the traverse example from pages 29 and 30, the angle adjustment has reduced the closure error to 0.0007', with ten times the precision of the original example.

additional HP-42S programming

Programming Examples and Techniques
Hewlett-Packard. Supplements the 42S owner's manual.

Synthetic Programming on the HP-42S
EduCALC Technical Notes (TN#24) by Richard Nelson.
Free (send a 25¢ stamp with your request) from

EduCALC Mail Store
27953 Cabot Road
Laguna Niguel, CA 92677

Individual solutions booklets (available summer 1989) by **D'Zign**. Titles will include:

Vertical Alignment | EDM Slope Staking | Topography
Alignment and Offsets | Spiral Curves | Urban Surveys

This program makes short work of calculating the layout of curves. Corrected chords are output if the curve is to be staked along an offset by the chord and deflection method, and adjusted tangent distances and offsets are output if you want to stake an offset by the tangent-offset method.

the subroutine, "STA"

```
01▶LBL "STA"
02 CF 29
03 FIX 00
04 STO 21
05 1E2
06 ÷
07 ENTER
08 IP
09 "␣"
10 ARCL ST X
11 -
12 ├"+"
13 FIX 03
14 1E2
15 x
16 10
17 X>Y?
18 ├"0"
19 ARCL ST Y
20 RCL 21
21 SF 29
22 FIX 04
23 RTN
```

There is one subroutine to be input for this program, shown to the left. Since it is not a routine you would normally call directly from the keyboard, it can go at the end portion of the program catalogue.

Go to "**INV**", enter the program mode, and then scroll upward to step 00 to begin typing in the program steps shown.

This subroutine changes a number, XXXX.xx, into the 'station' form, XX+XX.xx. This enhances the output and makes for one less chance to mistake a number for something else.

"Station 1,234.56" does not read as a station to most surveyors, while 12+34.56 is immediately recognized as a station even if it isn't labeled "station".

The step '1E2' (05 and 14) is input with the "**E**" key just to the left of the clear key. Stroke **E,2** and then the 'divide' (step 05) or 'times' (step 14) key to complete the entry.

editing "CURVE"

Since this program also uses "**CURVE**" as a subroutine, you have to do some quick editing.

Go to "CURVE", then go to step 147 and insert the two new steps shown to the right.

```
148 FS? 95
149 GTO "CLAY"
```

The new steps inserted into "**CURVE**" will send program execution back to the layout program after the curve data has been calculated.

The main program, "Curve **LA**Yout", should be near the top of program memory, for easy access. Go to "**COMP**" or "**TRAN**", (whichever you input), and scroll upward to 00. Begin input of the program steps shown below.

```
01▶LBL "CLAY"
02 FS?C 95
03 GTO 00
04 CF 02
05 CF 03
06 CF 04
07 CF 06
08 CF 07
09 SF 08
10 SF 09
11 SF 95
12 GTO "CURVE"
13▶LBL 00
14 ADV
15 1
16 STO 16
17 0
18 STO 15
19 STO 09
20 RCL 11
21 →RAD
22 2
23 x
24 STO 05
25 CLMENU
26 "B.C."
27 KEY 1 GTO 01
28 "P.I."
29 KEY 3 GTO 03
30 "E.C."
31 KEY 5 GTO 05
32 MENU
33 "Known Station?"
34 PROMPT
35▶LBL 01
36 STO 06
37 RCL+ 20
38 STO 14
39 RCL 06
40 RCL+ 24
41 STO 08
42 GTO 02
43▶LBL 03
44 STO 14
45 RCL- 20
46 STO 06
47 RCL+ 24
48 STO 08
49 GTO 02
50▶LBL 05
51 STO 08
52 RCL- 24
53 STO 06
54 RCL+ 20
55 STO 14
56▶LBL 02
57 RCL 06
58 STO 07
59 XEQ "STA"
60 ⊢" B.C.␊"
61 AVIEW
62 RCL 14
63 XEQ "STA"
64 ⊢" P.I.␊"
65 AVIEW
66 RCL 08
67 XEQ "STA"
68 ⊢" E.C.␊"
69 AVIEW
70▶LBL 04
71 CLMENU
72 "∡&LC"
```

```
73 KEY 2 GTO A
74 "∡&SC"
75 KEY 4 GTO B
76 "T O/S"
77 KEY 6 GTO C
78 MENU
79 CLA
80 PROMPT
81▶LBL A
82 SF 03
83 GTO 10
84▶LBL B
85 CF 09
86 SF 03
87 GTO 10
88▶LBL C
89 CF 09
90 CF 08
91 SF 04
92▶LBL 10
93 CLMENU
94 "O/S"
95 KEY 1 GTO D
96 "INTVL"
97 KEY 3 GTO E
98 "STA"
99 KEY 5 GTO F
100 MENU
101 CLA
102 PROMPT
103▶LBL F
104 CLA
105 XEQ "STA"
106 AVIEW
107 RCL- 07
108▶LBL 08
109 RCL 08
110 RCL- 07
111 X<>Y
112▶LBL 10
113 SF 07
114▶LBL 09
115 STO+ 07
116 XEQ 02
117 RCL 07
118 FC? 07
119 XEQ 20
120 RCL- 06
121 XEQ 02
122 RCL 10
123 FS? 03
124 GTO 03
125 FS? 04
126 GTO 04
127▶LBL 02
128 RCL÷ 05
129 STO 10
130 SIN
131 RCL× 11
132 2
133 ×
134 RCL× 16
135 RTN
136▶LBL 03
137 FIX 03
138 →HMS
139 "Chd = "
140 FC? 09
141 ARCL ST Z
142 FS? 09
143 ARCL ST Y
144 AVIEW
145 "Def.∡ = "
146 FS? 05
147 +/-
148 0
149 X<>Y
150 X<Y?
151 360
152 HMS+
153 XEQ "DMS"
154 AVIEW
155 CLA
156 FS? 02
157 GTO 15
158 GTO 10
159▶LBL 04
160 FIX 03
161 X<>Y
162 →REC
```

```
163 X<>Y
164 FS? 05
165 +/-
166 RCL 15
167 +
168 X<>Y
169 "Dist = "
170 ARCL ST X
171 ⊢"ч0/S = "
172 ARCL ST Y
173 AVIEW
174 FS? 02
175 GTO 15
176 GTO 10
177▶LBL E
178 STO 09
179 SF 02
180 RTN
181▶LBL D
182 STO 15
183 FIX 02
184 X<0?
185 SF 06
186 FC? 05
187 +/-
188 RCL+ 11
189 RCL÷ 11
190 STO 16
191 RCL 15
192 ABS
193 "Offset = "
194 ARCL ST X
195 FC? 06
196 ⊢"' RT"
197 FS? 06
198 ⊢"' LT"
199 AVIEW
200 ADV
201 FIX 04
202 GTO 10
203▶LBL 15
204 RCL 08
205 RCL 09
206 RCL+ 07
207 X<Y?
208 GTO F
209 ADV
210 0
211 STO 09
212 CF 02
213 GTO 10
214 END
```

To use the curve layout program stroke **XEQ**, then the key which corresponds to "**CLAY**". This brings up the first prompt bar DELTA R L T C D.

1. Input the two known parts of the curve, the same as when you use the curve calculation program, "**CURVE**"

Output will be the curve data for the curve, through the output of the chord.

Known Station?
B.C. P.I. E.C.

2. Input the station at the B.C., P.I. or E.C. and stroke the key which corresponds to the station input.

Output will be the stations of the B.C., P.I. and E.C., followed by the prompt bar for selection of the type of layout you want, X&LC X&SC T O/S.

3a. For deflection and LONG chord, X&LC

3b. For deflection and SHORT chord, X&SC

3c. For tangent-offset solution, stroke T O/S

O/S INTVL STA

4. If the curve is to be staked along an offset to centerline, input the offset distance (stroke the **+/-** key if the offset is to the left of centerline) and stroke O/S

5. If you want to calculate stations at constant intervals, input the interval distance and then stroke INTVL

6. Input the first station for which you want a solution, stroke **STA**

Output will be the station and either the deflection/chord or tangent/offset solution to the station.

If you have selected an interval, the program will continue through the curve without further input. When the interval stations have been completed the program will again display the prompt bar, allowing you to input additional stations which were not on even interval stations (inlet locations, etc.).

The example shown to the right is the solution for a curve with a radius of 500.00' and a central angle of 30°00'00", solved for deflection and long chord, along an offset of 10.00' left.

An additional station at 11+61.27 was calculated for the location of the center-line of a driveway.

As always, if you are not using a printer continue to stroke the **R/S** key for additional output or the next prompt.

You might want to try the same curve to see if you get these answers. If you don't, you should re-proof the program listing for incorrect steps.

```
    R = 500.0000
   D= 11°27'33.0"

DELTA= 30°00'00.0"
    L= 261.7994
    T= 133.9746
    C= 258.8190

10+00.000 B.C.

11+33.975 P.I.

12+61.799 E.C.

Offset = 10.00' LT

10+50.000
Chd = 50.979
Def.∡ = 2°51'53.2"

11+00.000
Chd = 101.830
Def.∡ = 5°43'46.5"

11+50.000
Chd = 152.427
Def.∡ = 8°35'39.7"

12+00.000
Chd = 202.643
Def.∡ = 11°27'33.0"

12+50.000
Chd = 252.352
Def.∡ = 14°19'26.2"

11+61.270
Chd = 163.783
Def.∡ = 9°14'24.3"
```

Coordinate transformation, or bearing rotation, is used to change a traverse from one "grid system" to another. A field traverse may be run without knowing the real basis of bearings, by beginning with an assumed bearing.

All of the normal adjustments can be made and the traverse worked with, new points calculated within the traverse, and so on.

When a basis of bearings becomes known (we finally got the description from the client), the bearings of the traverse may be rotated to match the "deed" bearings.

angle convention

The difference between the assumed (old) bearing along a known line and the deed (new) bearing of the same line is the ROTATION ANGLE. This program accepts the rotation angle as POSITIVE FOR CLOCKWISE and NEGATIVE FOR COUNTERCLOCKWISE.

Except when "blocks" of coordinates have been run, you have the option of reviewing or restoring the coordinates to the old system at any time. This program also allows two types of output, with or without the rotated courses also being shown, and the courses may be output as either azimuths or bearings.

input options

There are two types of setup input possible; **if the rotation angle is known** or **if two points in each system are known.** The instructions for input of the required information for each of the systems is shown on the following pages. Once the information has been input for either system, the solution steps for transformation of the points are the same for both systems (see solutions).

renumber option

A unique feature within this program allows you to transform points and renumber them at the same time, leaving the original coordinates as they were, in the old system. This can be handy for calculating the location of similar buildings on different lots.

rotation angle known

Call the program up by stroking

Rotation ∡ = ?

1. Input the rotation angle, in degrees, minutes and seconds. If the angle is counterclockwise, change sign with the **+/-** key, then stroke **R/S**

Scale Factor?

2. If the scale factor is 1:1, it is not necessary to input anything. If it is <u>not</u> 1, input the new factor before stroking **R/S**

Rotation Pt#?

3. Input the point number of the rotation point, stroke **R/S**

OUTPUT will be the point number and coordinates (if the printer is being used, "Rotating @ #x" will be output also).

New Coord's?

YES NO

4a. If the coordinates of the rotation point are the same in the new system as in the old, stroke NO

4b. If there are different coordinates for this point in the new system, stroke YES

If the answer to the last prompt was "yes" you will also receive the following prompts:

N =?

4b-1. Input the new north-coordinate, stroke **R/S**

E =?

4b-2. Input the new east-coordinate, stroke **R/S**

Next Point?

NEW | →OLD | BLK | INV→R | INV→B | RE #

Stroking any key (beginning input of the next point number) shows the prompt bar.

GO TO "OPTIONS"

two points in each system known

1. When the prompt, **Rotation ∡ =?**, is displayed, no input is necessary. Just stroke **R/S**

Rotation Pt#?

2. Input the point number of the pivot point, stroke **R/S**

New Coord's?

YES | NO | | | |

3. If the rotation point will have new coordinates in the new system stroke **YES**. If the coordinates are the same in both systems, stroke **NO**.

If the answer was "yes" you will receive the following prompts:

N =?

4a. Input the new north-coordinate, stroke R/S

E =?

4b. Input the new east-coordinate, stroke R/S

Second Point?

5. Input the point number of the second known point and stroke R/S

NEW X **N ↑ E**

6a. Input the new north-coordinate of the second point and stroke ENTER

6b. Input the new east-coordinate of the second point and stroke R/S

Next Point?

NEW →OLD BLK INV→B INV→A RE #

Stroking any key will bring up the prompt bar.

OPTIONS

At this point in the program you decide on the output you want. This can be just the new coordinates for the transformed points, or you can also calculate the directions and distances between the new coordinates.

1. If you want the distances and BEARINGS output, stroke INV→B

2. If you want the distances and AZIMUTHS output, stroke INV→A

You can transform a BLOCK of coordinates automatically, and use either of the above options at the same time.

1. After selection of the output type (above) input the block 'code', **AAA.bbb**, where AAA is the first point number and bbb is the second (three-digit input for the second number). Stroke [illegible]

Type?

2. With practice you can reverse some of the stored points by stroking →OLD, but for normal transformation to the new system stroke NEW

OUTPUT will be automatic, transforming all of the points within the block before stopping.

You can change the point numbers as the points are rotated if you stroke [illegible]. This option will NOT work if you are rotating a BLOCK of coordinates, and requires input of each point. You may not return to the "old" system while using this option either.

After input of the OLD point number in response to the "Next Point?" prompt, and stroking NEW, you will receive the prompt

NEW PT#?

1. Input the number you want the point to have in the NEW system and stroke R/S

OUTPUT will be the transformed coordinates under the new point number.

SOLUTIONS

Next Point?

1. Input the point number of the point which you wish to transform. (NOTE: the point numbers input for setup <u>have not yet</u> been transformed.

2a. To transform **from the "old" system to the "new" system**, input the point number and stroke

2b. To transform **from "new" system to "old"** stroke

The "old" system coordinates will be displayed (or printed out) but are not stored again under the point number.

Store?

2b-1. Answer **NO** if you do not want to restore the point with the "old" coordinates, **YES** if you do.

If your answer is yes, the message "STORED" will be seen, briefly, in the display.

Next Point?

Continue repeating solution step 2a and/or 2b as needed.

the program listing

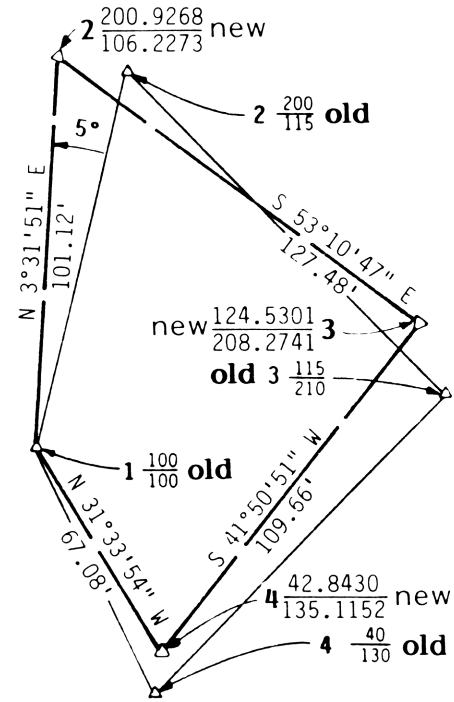
This program uses 700 bytes of programming. This is the equivalent of about 39 point numbers. The program should be input at a place where it is easy to find when you stroke the **XEQ** key, so why not just put it in at the **.END.**? Stroke **shift, XEQ .END.** and enter program mode.

```
00 { 700-Byte Prgm }
01▶LBL "C.ROT"
02 XEQ "F▪▪"
03 XEQ "CL"
04 SF 21
05 SF 10
06 1
07 STO 05
08 CF 22
```

```
09 "Rotation ∡ = ?ɥ"
10 PROMPT
11 CLA
12 FC? 22
13 GTO 02
14 +/-
15 →HR
16 STO 04
17 CF 22
18 "Scale Factor?ɥ"
19 PROMPT
20 CLA
21 FS? 22
22 STO 05
23 XEQ 03
24 XEQ 14
25 GTO a
26 RTN
27▶LBL 02
28 XEQ 03
29 XEQ 14
30 "Second Point?ɥ"
31 PROMPT
32 CLA
33 XEQ "POUT"
34 RCL- 01
35 X<>Y
36 RCL- 00
37 →POL
38 1/X
39 STO 05
40 X<>Y
41 XEQ 05
42 STO 04
43 "NEW "
44 XEQ 07
45 ⊢"N ↑ Eɥ"
46 FIX 04
47 PROMPT
48 RCL- 03
49 X<>Y
50 RCL- 02
51 →POL
52 STO× 05
53 X<>Y
54 XEQ 05
55 STO- 04
56 GTO a
57▶LBL 00
58 FC? 03
59 RTN
60 ISG 13
61 RTN
62 CF 03
63 GTO a
64▶LBL 14
65 CF 21
66 " New Coord's?"
67 AVIEW
68 SF 21
69 CLMENU
70 XEQ "YN"
71 FC? 10
72 RTN
73 R↓
74 GTO 08
75▶LBL A
76 SF 01
77 CF 02
78 XEQ 00
79 FS? 03
80 RCL 13
81 STO 16
82 XEQ "POUT"
83 RCL- 01
84 X<>Y
85 RCL- 00
86 →POL
87 RCL× 05
88 X<>Y
89 RCL- 04
90 X<>Y
91 →REC
92 RCL+ 02
93 X<>Y
94 RCL+ 03
95 CLA
96 FS? 04
97 XEQ 20
98 XEQ "IN"
```

```
99 FS? 09
100 XEQ 09
101 XEQ "OUT"
102 STO 08
103 X<>Y
104 STO 07
105 X<>Y
106 CF 10
107 " New"
108 FS? 55
109 AVIEW
110 XEQ "C↑"
111▶LBL a
112 FS? 03
113 GTO 01
114 XEQ 04
115 "Next Point?"
116 PROMPT
117 CLA
118 FS? 02
119 GTO B
120 FS? 01
121 GTO A
122 STOP
123▶LBL B
124 CF 01
125 SF 02
126 XEQ 00
127 FS? 03
128 RCL 13
129 "OLD "
130 FS? 55
131 AVIEW
132 STO 16
133 XEQ "POUT"
134 RCL- 03
135 X<>Y
136 RCL- 02
137 →POL
138 RCL÷ 05
139 X<>Y
140 RCL+ 04
141 X<>Y
142 →REC
143 RCL+ 00
144 X<>Y
145 RCL+ 01
146 CF 10
147 XEQ "C↑"
148 ADV
149 CF 21
150 "    STORE?"
151 AVIEW
152 SF 21
153 CLMENU
154 XEQ "YN"
155 FC? 10
156 CLX
157 FC? 10
158 GTO a
159 RCL 16
160 STO 13
161 R↓
162 XEQ "IN"
163 CF 21
164 "  STORED␊"
165 AVIEW
166 SF 21
167 GTO a
168 STOP
169▶LBL 03
170 "Rotation Pt#?␊"
171 PROMPT
172 CLA
173 CF 29
174 FIX 00
175 "Rotating @ #"
176 ARCL ST X
177 ⊢"␊"
178 FS? 55
179 AVIEW
180 XEQ "POUT"
181 STO 01
182 STO 07
183 STO 03
184 R↓
185 STO 00
186 STO 08
187 STO 02
188 FS? 55
```

```
XEQ 11
RTN
▶LBL 06
SF 03
1
-
STO 13
CF 21
"  Type?"
AVIEW
SF 21
XEQ 04
RTN
▶LBL 01
FS? 02
GTO B
FS? 01
GTO A
▶LBL 10
SF 09
CF 00
GTO 04
▶LBL 12
SF 09
SF 00
▶LBL 04
CLMENU
"NEW"
KEY 1 GTO A
"→OLD"
KEY 2 GTO B
"BLK"
KEY 3 GTO 06
"INV•A"
KEY 4 GTO 10
"INV•B"
KEY 5 GTO 12
"RE #"
KEY 6 GTO 16
MENU
FS? 03
GTO 15
RTN
▶LBL 08
XEQ "PN"
STO 03
STO 07
R↓
STO 02
STO 08
RTN
▶LBL 15
FS? 02
GTO B
FS? 01
GTO A
RTN
▶LBL 05
X>0?
RTN
360
+
RTN
▶LBL 07
FIX 00
ARCL 13
⊢"  "
RTN
▶LBL 09
XEQ "OUT"
XEQ "INV"
RTN
▶LBL 11
R↑
CF 10
XEQ "C↑"
ADV
RTN
▶LBL 16
SF 04
RTN
▶LBL 20
" NEW PT#?"
PROMPT
STO 13
R↓
.END.
```



The small traverse shown above will be used for the keystroke examples. Before beginning use "**LOAD**" (or "**PIN**") to input the coordinates in the "old" system, so that they are in memory.

In this first example we will rotate the bearings 5° to the left. We will also use the **auto–inverse** option to output the new bearings and distances as the points are rotated.

Stroke

XEQ

prompt: **Rotation ∡= ?**

keystroke:

5 +/- R/S

prompt: **Scale Factor?**

keystroke:

R/S

prompt: **Rotation Pt#?**

keystrokes:

1 R/S

output: Rotating @ #1

#1 N= 100.0000
E= 100.0000

prompt: **New Coord's?**

YES NO

keystroke: NO

prompt: **Next Point?**

keystrokes:

[INVRS] [2] [NEW]

output:

```
N 3°31'50.8" E
HD = 101.1187

New
#2   N= 200.9268
     E= 106.2273
```

prompt: **Next Point?**

keystrokes:

[3] [NEW]

output:

```
S 53°10'47.4" E
HD = 127.4755

New
#3   N= 124.5301
     E= 208.2741
```

prompt: **Next Point?**

keystrokes:

[4] [NEW]

output:

```
S 41°50'51.4" W
HD = 109.6586

New
#4   N= 42.8430
     E= 135.1152
```

prompt: **Next Point?**

keystrokes:

[1] [NEW]

output:

```
N 31°33'54.2" W
HD = 67.0820

New
#1   N= 100.0000
     E= 100.0000
```

prompt: **Next Point?**

As an example of how the points may be restored to the "old" system, stroke

[2] [→OLD]

output:

```
OLD
#2   N= 200.0000
     E= 115.0000
```

prompt: **STORE?**

[YES] [NO]

keystroke:

[YES]

. . . . **STORED** . . .

prompt: **Next Point?**

Go ahead and change points #3 and #4 back to the old system, and you'll be ready to try the second example without having to re-input the coordinates.

This is an example of the type where the coordinates of two points are known in each system, and uses "new" coordinates of 200/200 for #1 and 300/215 for #2. Run this one with AZIMUTH inverse.

keystrokes:

[XEQ] [CROT]

prompt: **Rotation ∡ = ?**

keystroke:

[R/S]

prompt: **Rotation Pt#?**

keystrokes:

1 R/S

output:

```
Rotating @ #1

#1  N= 100.0000
    E= 100.0000
```

prompt: **New Coord's?**

YES NO

keystroke:

YES

prompt: **N =?**

keystrokes:

2 0 0 R/S

prompt: **E =?**

keystrokes:

2 0 0 R/S

prompt: **Second Point?**

keystrokes:

2 R/S

prompt: **NEW 2 N ↑ E**

Note that for this prompt the northing is ENTERED, the easting is input, and then RUN is stroked.

keystrokes:

3 0 0 ENTER

2 1 5 R/S

prompt: **Next Point?**

keystrokes:

INVRT 2 NEXT

output:

```
HD = 101.1187

New
#2  N= 300.0000
    E= 215.0000
```

When using this type (two points known) the distance is output as a check, but it is assumed that the azimuth was known, since you knew the new coordinates of both points.

prompt: **Next Point?**

keystrokes:

3 NEXT

output:

```
AZ = 131°49'12.6"
HD = 127.4755

New
#3  N= 215.0000
    E= 310.0000
```

prompt: **Next Point?**

keystrokes:

4 NEXT

output:

```
AZ = 226°50'51.4"
HD = 109.6586

New
#4  N= 140.0000
    E= 230.0000
```

prompt: **Next Point?**

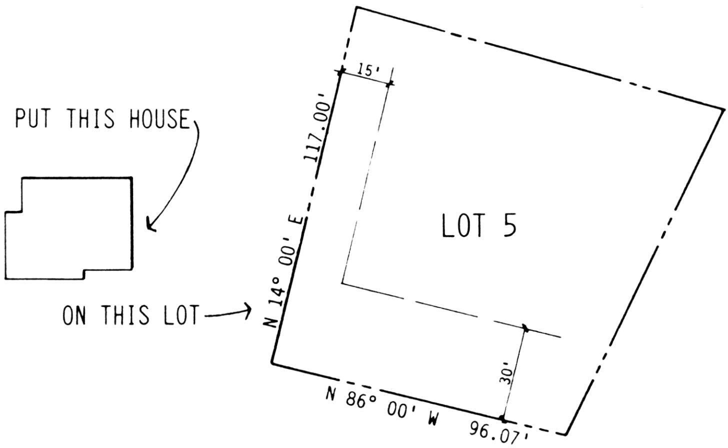
keystrokes:

1 NEXT

output:

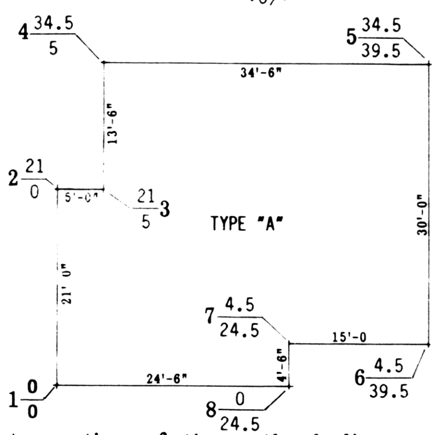
```
AZ = 333°26'05.8"
HD = 67.0820

New
#1  N= 200.0000
    E= 200.0000
```

If this problem looks familiar, you'll be happy to know that this program has a quick solution to it.



Quite often the client wants to put type "A" houses on lots 3, 5, 8 and 11 . . . type "B" on lots ....

Use the building's dimensions to set "dummy" coordinates on each of the corners, as shown to the right, and then rotate them into position.



Pre-calculate the intersection of the setback lines, you already know the bearing of the lot line. With this program you can store dummy points for the different types of buildings being used, and when you rotate them, use the re-number feature to give each lot an individual set of coordinates for layout. And, you will still have the original dummy points to use for the next lot.

Input the dummy points from page 75, and using the "calculated" coordinates for the setback corner, shown to the right, we'll try it out.

Stroke

[XEQ] [C.ROT]

prompt: **Rotation ∡ =?**

keystrokes:

[1] [4] [R/S]

prompt: **Scale Factor?**

keystroke:

[R/S]

prompt: **Rotation Pt#?**

keystrokes:

[1] [R/S]

output: Rotating @ #1

#1 N= 0.0000
E= 0.0000

prompt: **New Coord's?**

[YES] [NO]

keystroke:

[YES]

prompt: **N =?**

keystrokes:

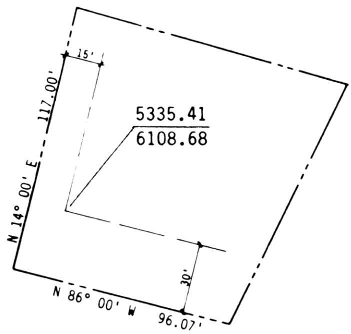
[5] [3] [3] [5] [.] [4] [1] [R/S]

prompt: **E =?**

keystrokes:

[6] [1] [0] [8] [.] [6] [8] [R/S]

prompt: **Next Point?**



keystrokes:

[REF:] [1] [NEW]

prompt: **NEW PT#?**

keystrokes:

[2] [1] [R/S]

output: New
#21 N= 5335.4100
E= 6108.6800

prompt: **Next Point?**

keystrokes:

[2] [NEW]

prompt: **NEW PT#?**

keystrokes:

[2] [2] [R/S]

output:

New
#22 N= 5355.7862
E= 6113.7604

Continue through point number 8=28. You can use "**DUMP**" to check that the original points are still intact.

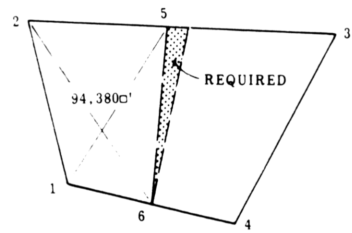
PREDETERMINED AREAS

There are two types of solution routines for solving for a predetermined amount of area. These are used to "part the land", or cut off a specific quantity of property from a larger parcel.

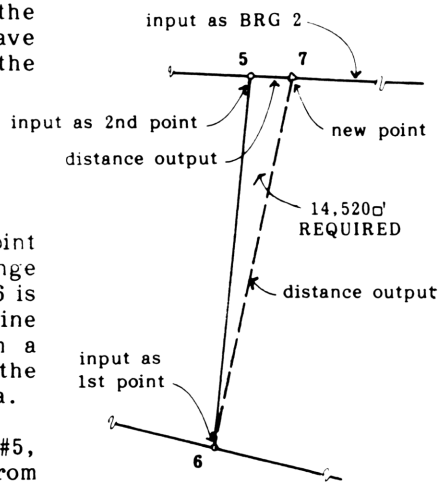
The illustration (below) shows a typical use of the type of solution called **Line Through A Point**. A parcel boundary has been run (points 1 through 4) and the area calculated.

To divide it into two equal parcels, first set two arbitrary points, 5 and 6 (6 is the half-way point along the base 1-4) and calculate the area of one parcel.

In this case, the area of the parcel 1, 2, 5, 6, 1 was run, and the shaded portion represents the remainder needed to have each parcel contain ½ of the original area.



TOTAL AREA
= 217,800□'

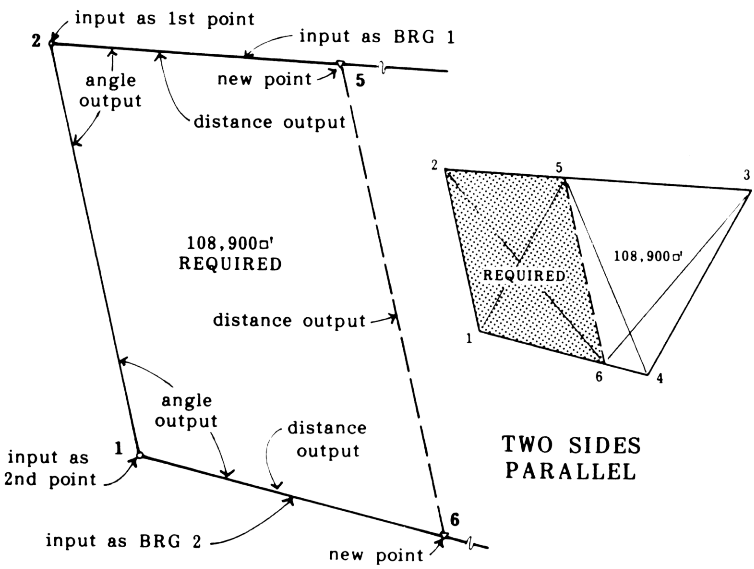


In this example, point number 6 is used as hinge point, and a line from 6 is intersected with the line from #5 to #3 to form a triangle that contains the required amount of area.

Input would be point #5, the bearing of the line from 5 to 3, point 6 and the area needed.

Two Sides Parallel is the second of the routines, and defines the boundary of a trapezoidal parcel whose area is predetermined.

If the requirement is that the area be one-half of the original parcel, but with the dividing line parallel to one of the lines of the original parcel, we would use this type of solution.



TWO SIDES PARALLEL

Input required for this method would be point #2, the bearing of line 2-3, point #1, the bearing of line 1-4, and the required area. The bearings are input in the direction **away** from the known points.

the trade

Adding this program to your set will replace about 30 points in storage usage. In addition to the main program there are a couple of short subroutines to be added. We'll start with those.

```
125 GTO 02
126▶LBL "BR"
127 "⊢Brg "
128 XEQ 01
129 GTO "A1"
130▶LBL "DS"
131 CF 29
132 FIX 00
133 ARCL 13
134 ⊢" Dist"
135 GTO 02
136▶LBL 01
137 CF 29
138 FIX 00
139 ARCL 13
140▶LBL 02
```

If you are going to add "**PRE-A**" to your program file, the program steps shown to the left are the ones that will be doing the labeling of the output.

A lot of program steps can be saved by inserting them into the subroutine stack, because they use the same processing labels as some of the other subroutines.

Go to program "**A0**", enter program mode, and scroll down two steps so that the pointer is at 124 SF 85. Begin typing in the new steps (shown to the left).

After they have been entered, scroll down once. Step 141 should be XEQ 06.

It wouldn't hurt to proof-read the new steps. Make sure that step 134 is "append, space, Dist".

the program listing

This program can fit in nicely between "**INT-X**" and "**CURVE**" in the main menu.

```
00 { 532-Byte Prgm }
01▶LBL "PRE-A"
02 XEQ "F▪"
03 XEQ "CL"
04 CLMENU
05 CF 21
06 "Triangle?"
07 AVIEW
08 SF 21
09 XEQ "YN"
10 FS? 10
11 GTO 03
12 SF 01
13▶LBL 03
14 SF 10
15 SF 00
16 "1st Pt#?⊢"
17 PROMPT
18 STO 15
19 XEQ "POUT"
20 STO 01
21 X<>Y
22 STO 00
23 FC? 01
24 GTO 05
25 XEQ "BR"
26 STO 02
27▶LBL 05
28 "2nd Pt#?⊢"
29 PROMPT
30 STO 16
31 1E3
32 ÷
33 RCL+ 15
34 XEQ "CODE"
35 →HR
36 STO 23
```

```
37 RCL 16
38 XEQ "POUT"
39 STO 04
40 X<>Y
41 STO 03
42 XEQ "BR"
43 STO 05
44 "Next Pt#?ч"
45 PROMPT
46 STO 17
47 1
48 +
49 STO 22
50▶LBL 08
51 "Req'd Area?ч"
52 PROMPT
53 STO 08
54 FS? 01
55 GTO 01
56 FC? 00
57 GTO 02
58 XEQ 00
59 RCL 05
60 -
61 STO 05
62▶LBL 02
63 RCL 08
64 RCL 05
65 SIN
66 ABS
67 RCL 06
68 x
69 ÷
70 2
71 x
72 STO 09
73 RCL 15
74 STO 13
75 RCL 05
76 RCL 06
77 →REC
78 RCL 09
79 -
80 +/-
81 →POL
82 X<>Y
83 RCL 05
84 +
85 COS
86 +/-
87 ACOS
88 RCL 23
89 X<>Y
90 -
91 →HMS
92 X<>Y
93 CF 10
94 XEQ 06
95 ARCL 15
96 ├"-"
97 ARCL 17
98 XEQ 04
99 XEQ 22
100 RCL 05
101 COS
102 ACOS
103 RCL 23
104 +
105 →HMS
106 RCL 09
107 XEQ 06
108 ARCL 17
109 ├"-"
110 ARCL 16
111 XEQ 04
112 XEQ 22
113 FC? 00
114 STOP
115 RCL 07
116 RCL 05
117 -
118 RCL 09
119 →REC
120 RCL 03
121 +
122 X<>Y
123 RCL 04
124 +
125 RCL 17
126 STO 13
```

```
127 R↓
128 XEQ "IN"
129 XEQ "OUT"
130 FIX 04
131 ADV
132 XEQ "C↑"
133 STOP
134▶LBL 00
135 360
136 RCL 01
137 RCL 04
138 -
139 RCL 00
140 RCL 03
141 -
142 →POL
143 STO 06
144 R↓
145 X<0?
146 +
147 STO 07
148 RTN
149▶LBL 01
150 FC? 00
151 GTO 02
152 XEQ 00
153 RCL- 05
154 COS
155 ACOS
156 STO 12
157 PI
158 →DEG
159 RCL 07
160 RCL- 02
161 COS
162 ACOS
163 -
164 STO 11
165▶LBL 02
166 RCL 06
167 X↑2
168 RCL 11
169 COS
170 LASTX
171 SIN
172 ÷
173 RCL 12
174 COS
175 LASTX
176 SIN
177 ÷
178 +
179 STO 10
180 RCL× 08
181 2
182 x
183 -
184 SQRT
185 STO 14
186 RCL+ 06
187 RCL 08
188 2
189 x
190 X<>Y
191 ÷
192 STO 09
193 RCL 11
194 SIN
195 ÷
196 STO 24
197 RCL 11
198 RCL 23
199 X<>Y
200 -
201 →HMS
202 X<>Y
203 XEQ 06
204 ARCL 15
205 ├"-"
206 ARCL 17
207 XEQ 04
208 XEQ 22
209 XEQ 07
210 RCL 09
211 RCL 12
212 SIN
213 ÷
214 STO 10
215 RCL 12
216 RCL 23
```

```
217 +              240 RCL 17         263 R↓
218 →HMS           241 STO 13         264 XEQ "A→B"
219 X<>Y           242 RCL 00         265 RTN
220 XEQ 06         243 RCL 01         266▸LBL 07
221 ARCL 22        244 XEQ "IN"       267 RCL 23
222 ├"-"           245 XEQ "OUT"      268 →HMS
223 ARCL 16        246 ADV            269 RCL 14
224 XEQ 04         247 XEQ "C↑"       270 XEQ 06
225 XEQ 22         248 ISG 13         271 ARCL 17
226 FC? 00         249 STO ST X       272 ├"-"
227 GTO 03         250 RCL 03         273 ARCL 22
228 RCL 02         251 RCL 04         274 XEQ 04
229 RCL 24         252 XEQ "IN"       275 XEQ 22
230 →REC           253 XEQ "OUT"      276 RTN
231 STO+ 00        254 ADV            277▸LBL 04
232 X<>Y           255 XEQ "C↑"       278 ASTO 13
233 STO+ 01        256 ADV            279 CLA
234 RCL 05         257 RTN            280 RTN
235 RCL 10         258▸LBL 22         281▸LBL 06
236 →REC           259 CF 10          282 CLA
237 STO+ 03        260 XEQ "DS"       283 FIX 00
238 X<>Y           261 ├" ↳"          284 CF 29
239 STO+ 04        262 AVIEW          285 END
```

choosing the type

The type of solution to be used is selected by your response to the first prompt:

Triangle?

If you want to use the LINE THROUGH A POINT routine, answer YES. If you want the TWO SIDES PARALLEL routine, stroke NO.

line through a point

1st Pt#?

1. Input the point number for the first point (the "hinge" point) and stroke

R/S

2nd Pt#?

2. Input the point number which represents the fixed point and stroke

R/S

Brg 2 = ?

3. Input the known bearing of the fixed line from the second point, and stroke

ENTER

Input the quadrant code in the direction away from the known point, then stroke

R/S

Next Pt#?

4. Input the UNUSED point number you want to assign to the intersection point, stroke

R/S

Req'd Area?

5. Input the required area in square feet (or square meters, etc.) and stroke

R/S

OUTPUT will be the distance and bearing from the hinge point to the new calculated point, then the distance and bearing along the known course from the new point to the second point, followed by the coordinates of the new (intersection) point.

EXIT

two sides parallel

1st Pt#?

1. Input the point number of the first point

R/S

Brg 1 = ?

2. Input the bearing from the first point

ENTER

Input the quadrant code, in the direction away from the point, and stroke

R/S

2nd Pt#?

3. Input the point number of the second fixed point and stroke

R/S

Brg 2 = ?

4. Input the bearing of the line radiating from the second fixed point and stroke

ENTER

Input the quadrant code, in the direction away from the point, and stroke

R/S

Next Pt#?

5. Input the next UNUSED point number. This point number will be assigned to the point along the line defined by bearing 1. The next highest point number will be assigned to the point along bearing 2. Stroke

R/S

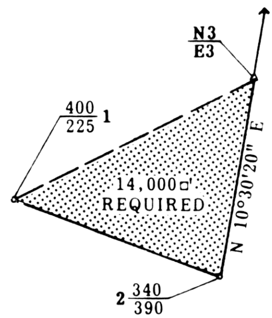
Req'd Area?

6. Input the required area and stroke

R/S

OUTPUT will be the distances and bearings of the three lines, followed by the coordinates of the two new points.

We will begin the keystroke examples with the **triangular**, or **line through a point** routine. The illustration to the right will be used for the example.



Input the coordinates for points #1 and #2 (from the example) with "**LOAD**" or "**PIN**" before beginning this example. Begin the program by stroking

[XEQ] [illegible]

prompt: **Triangle?**

[YES] [NO]

keystrokes:

[YES]

prompt: **1st Pt#?**

keystrokes:

[1] [R/S]

prompt: **2nd Pt#?**

keystrokes:

[2] [R/S]

prompt: **Brg 2 = ?**

keystrokes:

[1] [0] [.] [3] [0] [2] [ENTER]

[1] [R/S]

prompt: **Next Pt#?**

keystrokes:

[3] [R/S]

prompt: **Req'd Area?**

keystrokes:

[1] [4] [0] [0] [0] [R/S]

output:

```
1-3 Dist = 218.2180
N 63°01'37.7" E
3-2 Dist = 161.6871
S 10°30'20.0" W

#3  N= 498.9768
    E= 419.4805
```

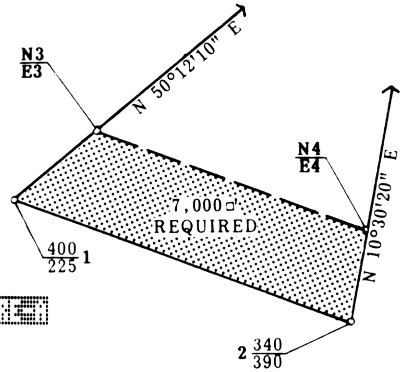
keystroke:

[EXIT]

It is always a good idea to run an inverse traverse to check the area.

The next example uses the same coordinates for points #1 and #2, so you won't have to input new coordinates.

For the example of the **two sides parallel** routine, we will use the illustration shown to the right.



Begin by stroking **XEQ** [illegible]

prompt: **Triangle?**

keystrokes:

NO

prompt: **1st Pt#?**

keystrokes:

1 **R/S**

prompt: **Brg 1 = ?**

keystrokes:

5 **0** **·** **1** **2** **1** **ENTER**

1 **R/S**

prompt: **2nd Pt#?**

keystrokes:

2 **R/S**

prompt: **Brg 2 = ?**

keystrokes:

1 **0** **·** **3** **0** **2** **ENTER**

1 **R/S**

prompt: **Next Pt#?**

keystrokes:

3 **R/S**

prompt: **Req'd Area?**

keystrokes:

7 **0** **0** **0** **R/S**

output:

```
1-3 Dist = 50.9226
N 50°12'10.0" E
3-4 Dist = 142.5946
S 70°01'00.8" E
4-2 Dist = 44.6113
S 10°30'20.0" W

 #3   N= 432.5941
      E= 264.1246

 #4   N= 383.8634
      E= 398.1340
```

EXIT

ultimate sizing

After you have input all of the programs you are planning to use, you can re-size the calculator to hold the maximum number of coordinate pairs. Each pair of coordinates occupies two registers, so you want to size with as many registers as you can.

1. Size to 0025.

2. Check the available memory (hold down the key under **MEM**, in CATALOG), and jot down the number.

   This is the number of **bytes** available. Divide by 9 (each register uses 9 bytes) and add 23 to this new number.

3. Re-size to the number you just calculated.

4. To calculate the number of points you can store, divide the number you jotted down by 18 and subtract 1.

adding more programs

If, at a later date, you decide to add more programs to what you now have in the calculator, re-size to 0025 before beginning, and then repeat the process above after you are done.

MORE HP42S PROGRAMS

We have a number of new booklets, and another book, of programs for the **HP42S** in the works! Here are some of the titles:

Vertical Alignment (booklet, $10.00)

Calculates CONTINUOUS vertical alignment without changing back and forth between Grade and Curve routines. Calculates vertical intersections, symmetrical or asymmetrical vertical curves. Solves for station when the elevation is known, or the station can be given, to calculate the elevation.

Spiral Curves (booklet. 00)

Calculate OUT OF PRINT .oordinates to any station, or offset to a ., within a spiral system. Options include .iate output, auto-inverse, or both.

Calculates intersections of the entrance or exit spiral with a circular curve or straight line.

Topography (booklet, $10.00)

This one turns your 42S into a manual data collector, complete with a labeling system that you can customize to suit the type of topo work you do. All shots are stored as finished data, by shot number, for later output. Choice of 3-D coordinates or Station-Offset-Elevation for the output.

EDM Slope Staking (booklet, $10.00)

Set up anywhere near an alignment and slope stake it. Sets slope stakes from the remote instrument location directly. Includes a three-point resection program for finding the instrument's location by either station-offset or coordinates. All data needed to mark the stake is output (or may be stored), and there is a subroutine for setting the reference stake.

Alignment/Offsets (booklet, $10.00)

Follows any alignment's circular curves and tangents, letting you calculate the coordinates or radial ties to any station or offset to a station. Coordinate output, auto-inverse, or both.

Triangle Solutions (booklet, $10.00)

The 42S version of the most complete triangle solutions program ever available. Solves with any of the following knowns: ASA SAA SAS SSA SSS Area-SS Area-AA Area-SA.

keep in touch

As always, we're anxious to make programs available to surveyors who need them. Make sure that we have your name and address on file so that we can send you the 42S Newsletters as they are published. These newsletters contain useful hints and additional programming.

D'Zign

The Most Commonly Asked Questions

The following questions and answers were compiled from the calls and letters we've received in the past 4+ years that we've been publishing solution books for the HP42S calculator, and are included here in the event that your question is one of them.

Q: *How do you type in the **END**?*
A: There are a number of ways . . . one easy way is to stroke [XEQ] [ENTER] and type it in, using the alpha keys. Because you stroked [XEQ] first, the calculator will recognize that this in not an alpha input, and substitute the actual function when you stroke [ENTER] again. You may input *any* function by this method.

You may also take advantage of the built-in *function catalog*, stroke [ ] [+] (catalog), and then the [FCN] menu key. You may scroll up or down with the [▲] or [▼] keys, and ***all*** of the calculator's functions are in there. When you reach the one you want, just stroke the key under the menu item.

Q: *How do you type in the **indirect** calls, such as step 138 in the CURVE program on page 10?*
A: The indirect calls are made by stroking [•]. In the case of the call above, first stroke [ ] [6] (flags), then [CF], to bring up the prompt CF__, then stroke [•]. Some of the indirect calls give a secondary prompt, requiring another [•].

Q: *How do I type in a **ARCL** command?*
A: Enter ***alpha mode*** before stroking [RCL] or [STO].

$25.00 U.S.

2173

ISBN 0-944889-14-X