JASON E. FOOSE, PS

the HP 35s calculator

A Field Surveyor's Companion

Part 2—A Few Inner Workings Leading Up to Traverse

I presented the article Two Nifty Programs that will make your HP 35S Calculator "Cry And Sing!" in the August 2014 issue of The American Surveyor. I scoured that thing for errors like a polecat in a Piggly Wiggly dumpster. While I'm ecstatic to learn that our readers are getting into the subject matter, I regret to inform you of an oversight on my part. In my "polar to rectangular" conversion example I included an extra ENTER *keystroke after keying in the distance (magnitude/modulus component) in both samples. This inadvertently places the distance in both Y-reg and X-reg rather than the intended Y-reg=angle X-reg=distance. Special thanks goes out to reader* ***William L. Meagher of WM Surveys, Inc. Ventura, California,*** *for the catch and a toast to our polecat of the month award winner!*

The Inner Workings

The development of these programs has been an evolutionary expansion of my personal knowledge regarding programming logic. The components of the traverse routine are among my earliest attempts at keystroke programming and appear rudimentary to me now. The catalyst of improvement is the application of hindsight to foresight; however I have elected to present the programs as originally written while placing the challenge of improvement in the readers' hands. Please email your suggestions for improvement to *rls43185@gmail.com*. I will do my best to incorporate your suggestions in print as we progress through the column.

You will need to refer to the August 2014 issue and install the Rectangular and Polar programs. Make sure that you label the "R" and "P" accordingly. Traverse relies on six subroutines or nested programs to function. Rectangular and Polar are two of those.

This Month's Programs

LBL C manipulates a distance in the Y-reg with an azimuth in the X-reg into a complex number. This program was slightly modified from the work by and shown with permission of Jeremy Dean. Jeremy's original program can be found at *sac-surveyors.org/node/22* and can run as a "stand alone" program. Thanks Jeremy for this great little ditty!

Key in the following instructions beginning with BRS R/S {PRGM} to open program mode. C or BRS R/S {PRGM} will exit program mode.

C001	LBL C	
C002	STO I	
C003	R▼	*note: the roll down stack key*
C004	STO D	
C005	RCL I	
C006	SIN	
C007	x	*note: multiply*
C008	STO X	
C009	R▼	
C010	RCL I	
C011	COS	
C012	RCL D	
C013	x	*note: multiply*
C014	0*i*0	
C015	+	
C016	0*i*0	
C017	RCL X	
C018	+	
C019	1θ90	
C020	x	*note: multiply*
C021	+	
C022	RTN	

Table 1: Example Data and Running The Program as a "stand alone"

KEYSTROKE STEPS	RESULTANT DISPLAY	ACTION
100	Y-reg : 0 or default value X-reg : 100.00	Load the distance into the stack. Note display is fix 2.
ENTER	Y-reg : 100.00 X-reg : 100.00	Advances value to Y-reg.
45 LYS 8{HMS→}	Y-reg : 100.00 X-reg : 45.00	Load azimuth in X-reg. Note DMS→ is normally required.
XEQ XEQ ENTER	Y-reg : 45.00 X-reg : 70.71 *i* 70.71	Executes program C. Northing is the real number and Easting is the imaginary number when entering 360° azimuth values. Cartesian values will be reversed.

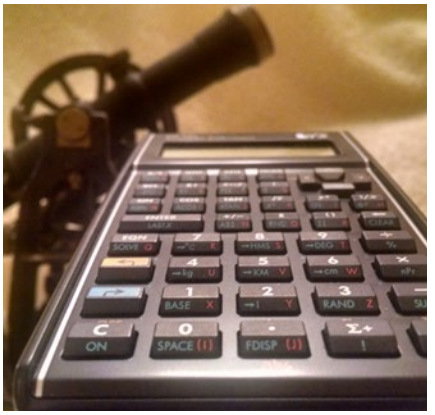
Jeremy's program is great and I have relied on it for several years without question. I recognized that the program radiated savviness beyond my stratosphere of insight. Revisiting my programs in conjunction with this series yielded a grin anointed with posterior knowledge and an alternative prescription for the same solution as follows:

C001	LBL C
C002	CF 2
C003	EQN ((SIN(REGX) x REGY) x (1θ90))+((COS(REGX) x REGY) x (1i0))
C004	RTN

Whereas Jeremy's original version efficiently follows the syntax and architecture of traditional keystroke programming, the alternate method demonstrates the true power of the HP 35s' ability to digest complicated equations. This example affords a shift in logic from traditional keystroke step programming to equation based programming. Besides the obvious reduction in programming lines, unlocking the power of this equation logic also patents the opportunity to expand our conventional wisdom beyond the established frontiers of RPN practice. After realizing the "wow" factor of this equation logic, I cut a big ol' slice of humble pie and understood why a rectangular/polar conversion key may not be such a big deal in the larger world of mathematics and handheld calculators. It took me a while but I finally caught up with HP's foresight.

Program LBL A is a dependent subroutine utility that manipulates and applies the traverse course to the point coordinates then returns to the traverse program at line T034.

A001	LBL A
A002	RCL E
A003	RCL N
A004	XEQ P001
A005	x<>y
A006	XEQ C001
A007	+
A008	R▲
A009	CL*x*
A010	x<>y
A011	RTN



©2014 JASON E. FOOSE

The program can be verified by storing the value of 100 in variables "E" and "N". Key the complex number 100*i*100 into the stack and ENTER then XEQ R/S (LBL A). The solution in the X-reg should be 200*i*200. This is simply random data to verify the mechanics of your input. There's no need to examine why this works at this point.

Program LBL J is a dependent subroutine utility that manipulates a complex number (coordinate pair) into its rectangular components and stores the values for later use.

J001	LBL J
J002	ENTER
J003	SF 10
J004	EQN "RCL PNT"
J005	FIX 0
J006	INPUT J
J007	x=0?
J008	GTO T011
J009	FIX 2
J010	VIEW (J)
J011	RCL (J)
J012	ARG
J013	LAST*x*
J014	ABS
J015	XEQ R001
J016	STO N
J017	x<>y
J018	STO E
J019	RTN

The program can be verified by entering coordinates for a point using program LBL H. The program will recall that point and display the rectangular components of that point in the X-reg and Y-reg accordingly. Again this is simply random data to verify the mechanics of your input.

Program LBL S is a dependent point storage subroutine.

S001	LBL S
S002	SF 10
S003	EQN "STORE PNT"
S004	FIX 0
S005	INPUT J
S006	FIX 2
S007	RCL C
S008	STO (J)
S009	VIEW (J)
S010	RCL (J)
S011	ARG
S012	LASTx
S013	ABS
S014	XEQ R001
S015	STO N
S016	x<>y
S017	STO E
S018	GTO T001
S019	RTN

The program can be verified by storing the complex number 99*i*120 into variable C. Run the program and enter point number 1 at the J? prompt. Press R/S and you should see the point number and the coordinate values in complex format. Press R/S again and the word "NONEXISTENT" should appear. This indicates that the program was correctly entered but LBL T (traverse) has yet to been defined. That's okay!

This installment's routines are intermediate building blocks for several programs. The routines that follow will be the actual working programs that produce tangible results. The format of the next few installments will include a programming routine, instructions, and sample data for use. The next installment will focus on the azimuth traverse routine and data entry conventions. Hopefully the information presented herein is clear and genuinely explanatory. Please do not hesitate to send any comments, concerns, questions, or criticism to *rls43185@gmail.com*. ■

Since 2012, **Jason Foose** has served as Arizona's Mohave County Surveyor. He is licensed in Arizona, Colorado, and Nevada and has been practicing since the Cleveland Browns 33rd anniversary of Superbowl celibacy.