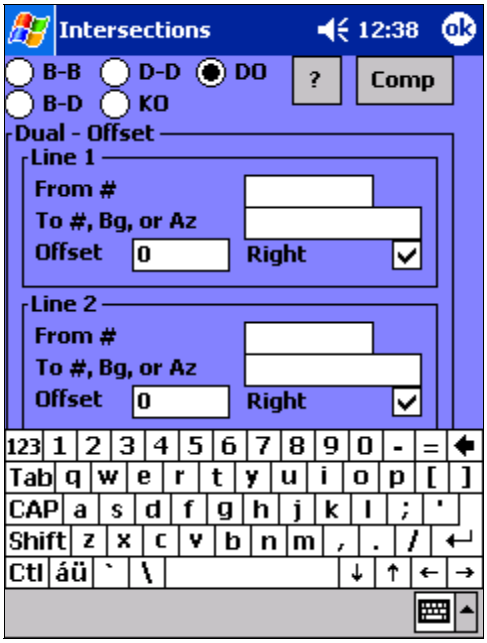
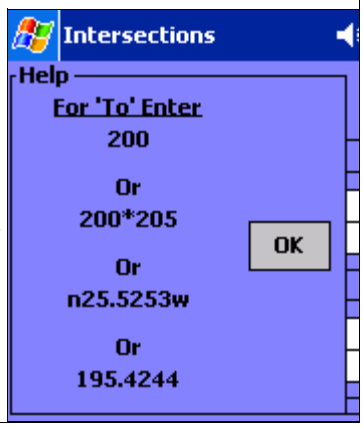
cEZ Intersections



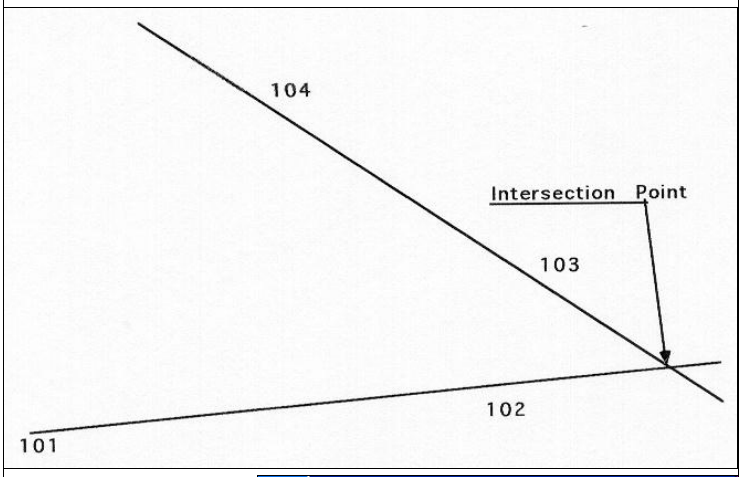
Prosurv cEZ contains 5 different intersection routines:

- Bearing-Bearing
- Bearing-Distance
- Distance-Distance
- Dual-Offset
- Known Offset

Tap the **?** button for hints on entering point numbers, bearings, and parallel lines when using the intersection routines.



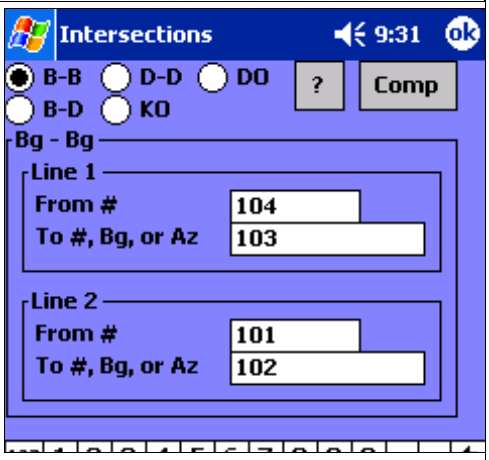
Bearing-Bearing Intersect



To compute the intersection of two lines, just enter two points from each line.

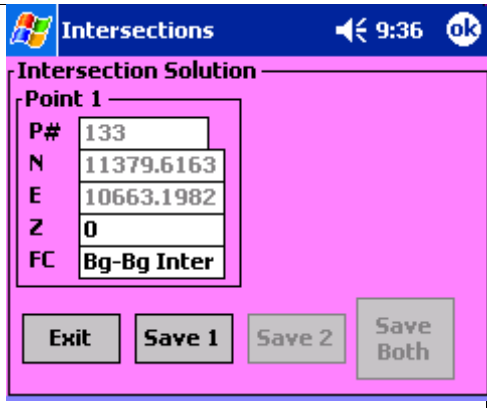
Tap the **Comp** button to compute the intersection point.

If **Instant Save** is On, the point will be stored immediately. If Instant Save is off, then the computed point will be displayed.

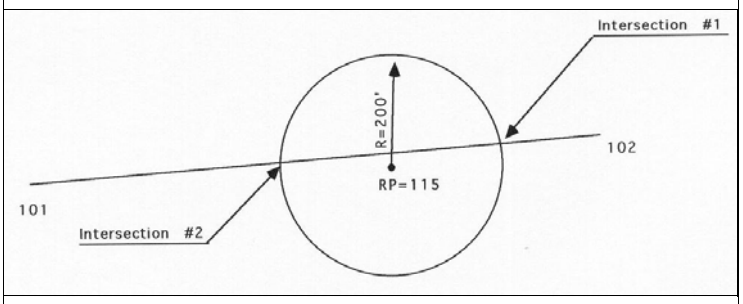


Instant Save is found in the **Points** button.

The intersection coordinate is displayed. You can enter an elevation and change the Feature Code. Tap **Save 1** to store the point.

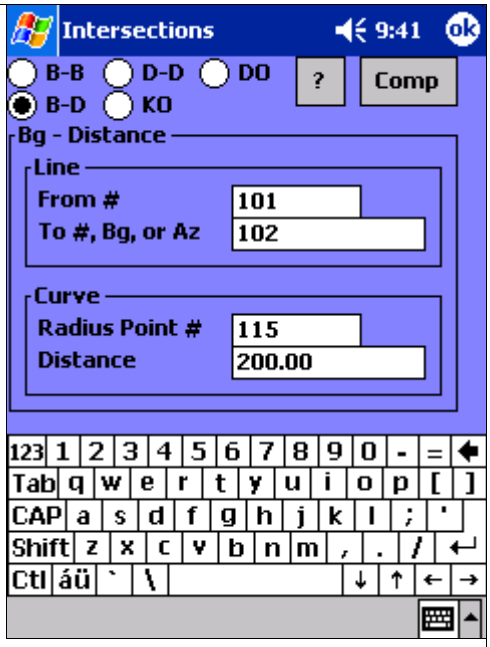


Bearing-Distance Intersection



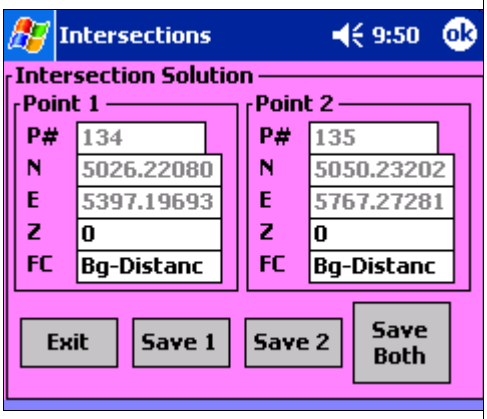
Whenever you compute a Bearing-Distance Intersection, there will always be two possible solutions. Prosurv cEZ computes and displays both solutions. You can Save #1, Save #2, or Save Both. If you're unsure as to which solution you need, simply Save Both, then check your plotting display to see which one is the one you want.

Tap the **Comp** button to compute the Bearing-Distance Intersection.

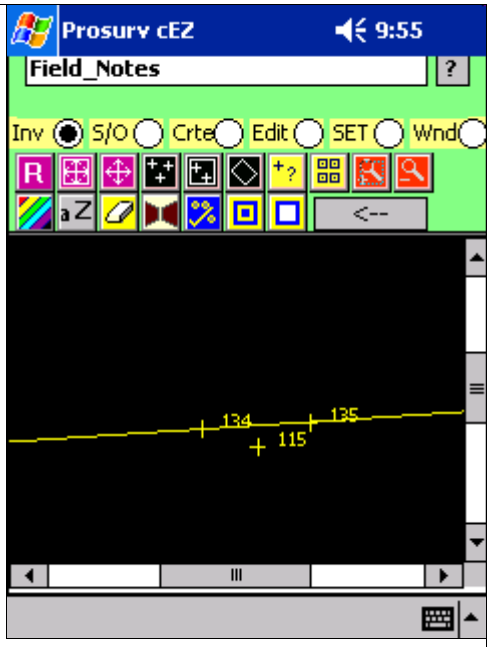


Both points are computed and displayed.

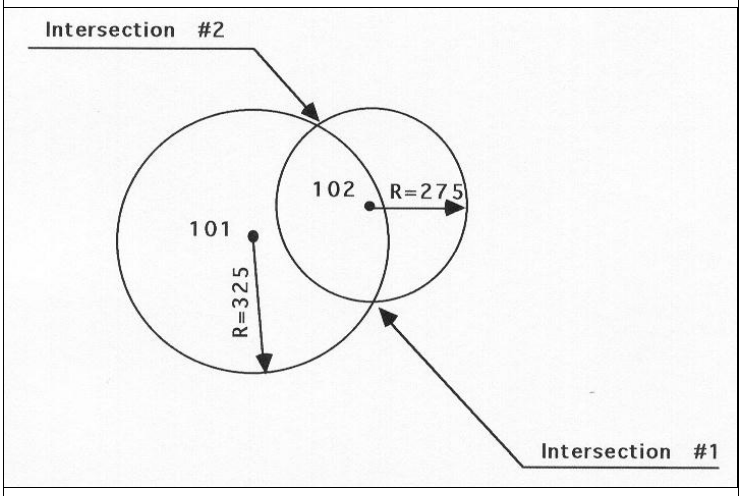
Tap **Save Both** to save both points. You could then view your plotting screen to determine which point you need.



Points 134 and 135 were saved and are graphically shown here.

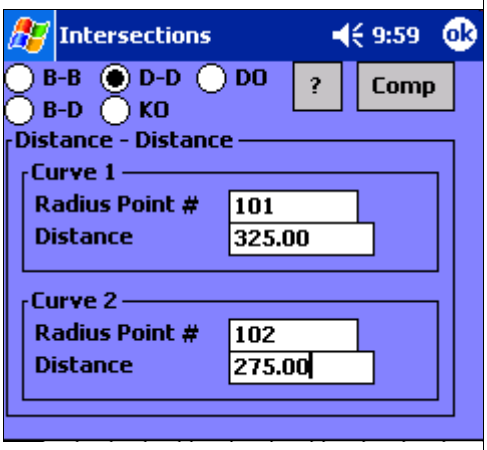


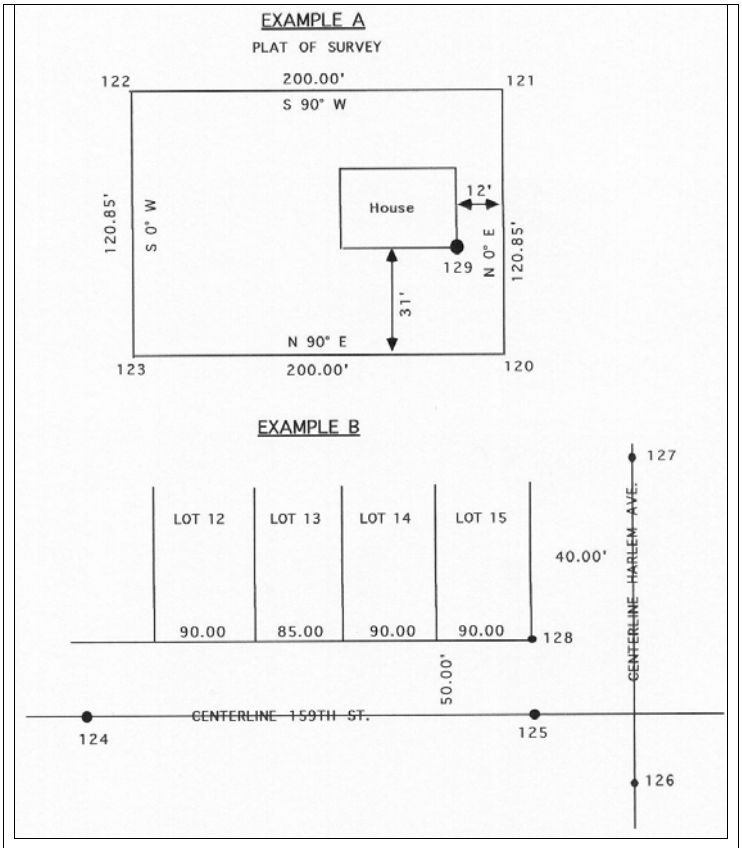
Distance-Distance Intersection



This routine also gives two solutions. You can Save #1, Save #2, or Save Both. If you save both, you can then view the points on the plotting screen to see which one you need.

Enter the point number and distance from each point. Tap the **Comp** button to compute the intersection.

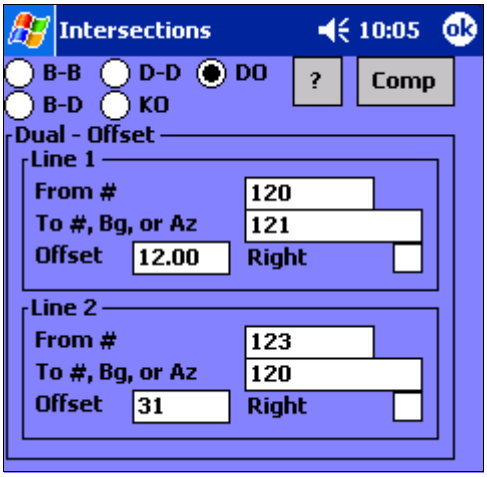




This function makes it very easy to create a point which represents the intersection of two different offset lines. This can be useful for computing a Block Corner when two centerlines are known. You can also use this function to create the corner of a house given two property lines and two setback lines (offsets from property lines). All that is needed are the two offset amounts and two points representing each line. Follow the examples below for a step-by-step guide to using the function.

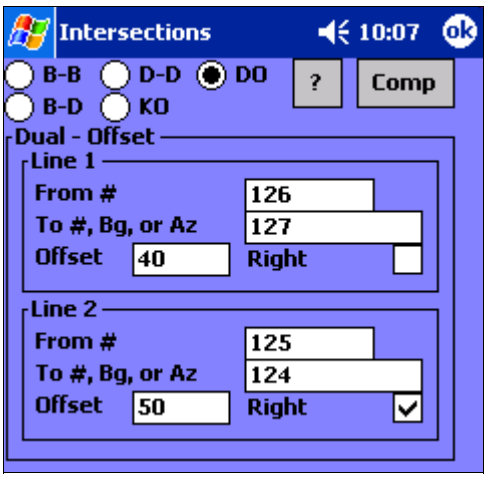
Data entry for example #1 is shown here.

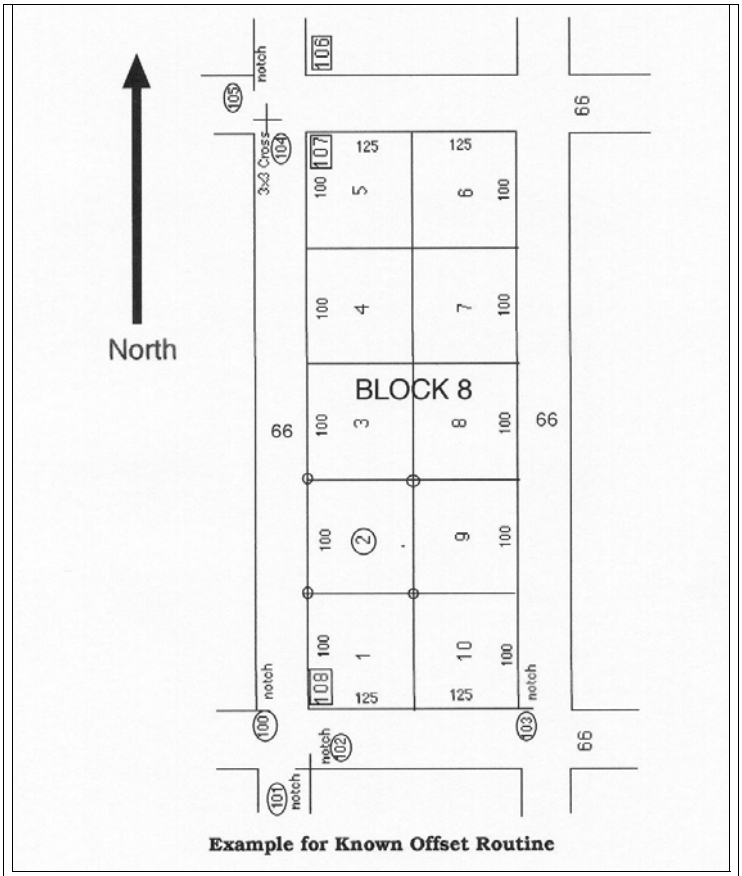
Tap the **Comp** button to compute the one intersecting point.



Data entry for example #2 is shown here.

Tap the **Comp** button to compute the one intersecting point.



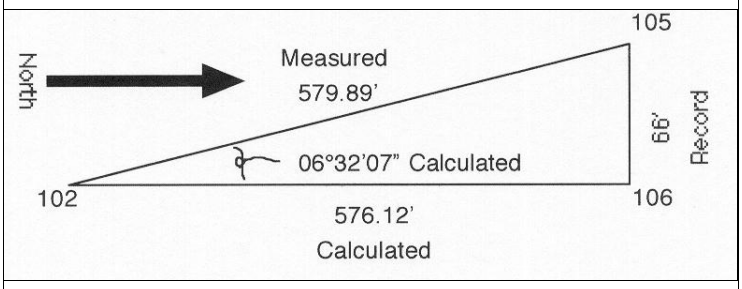


Example for Known Offset Routine

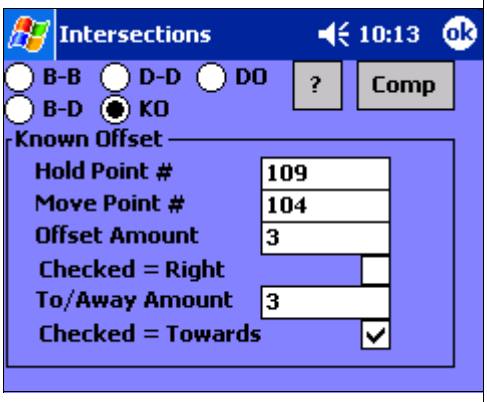
The **Known Offset** function is very useful when computing Block corners or any time when the shifting of an offset cross, iron pipe, rebar, brass cap, or notch is needed. Refer to the figure on the next page for the following example.

In computing the lot corners for Lot 2 in Block 8, it is desired that the Block Corners #107 and #108 be computed. This will be done by using the various found notches and crosses. The Block corner at 108 can be easily created by using the **Line-Line Intersect** routine. The first Line is 100-103. The second line would be at #102 sighting #105 pulled 66' East. This is where the Known Offset routine comes into play.

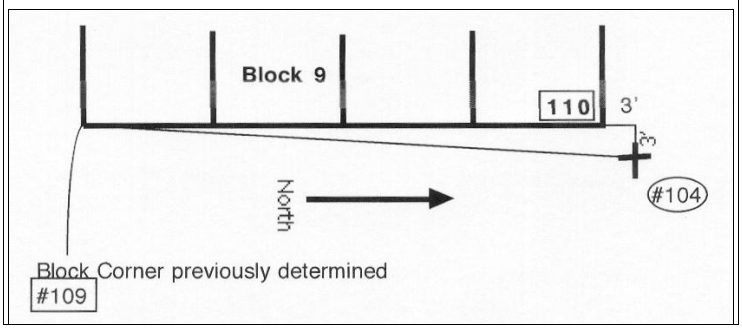
Known Offset will then create the coordinate (saved as #107) by calculating the angle and distance based on the triangle shown below:



In the same manner, point #104 (a found cross), can be 'moved' to the Block corner that it represents. It was determined in the field that it is a 3' by 3' offset cross. Therefore, you would enter the information as follows:



A new coordinate is created based on the triangle below:



The new block corner (saved as #110) is now created.

Note: Keep in mind that when using this function, the triangle which will be used will determine the outcome of the computed coordinate. For instance, to create #110 you would *not* use 'Hold #102' and specify '3 feet left' and '3 feet towards', since this would give a bogus answer. In other words, the point to be 'moved' should end up on a baseline with the point held. Also note that the point 'moved' is unaffected and its coordinate value is *not* changed.