FindLevels sets the following variables:

`V_flag`	0: *maxLevels* level crossings were found. 1: At least one but less than *maxLevels* level crossings were found. 2: No level crossings were found.
`V_LevelsFound`	Number of level crossings found.

Example

```
Macro FindLevelsExample()
   // Sample data
   Make/O/N=1024 peaks;SetScale/P x,0,0.001,"" peaks // 1ms sampling
   peaks=(exp(sawtooth(-(x-1)*300/pi)^3)-1)/(exp(1)-1)
   SetRandomSeed 0; peaks += gnoise(0.05)
   // locate rising and falling x crossings of y = 0.5
   Variable minDX=20*deltax(peaks) // min 20 samples between crossings
   Variable level = 0.5 // peaks y range nominally 0..1
   FindLevels/Q/D=risingEdges/EDGE=1/M=(minDX) peaks, level
   FindLevels/Q/D=fallingEdges/EDGE=2/M=(minDX) peaks, level

   // Indicate found edges
   Duplicate/O risingEdges, risingYs; risingYs = peaks(risingEdges[p])
   Duplicate/O fallingEdges, fallingYs; fallingYs = peaks(fallingEdges[p])
   // show level
   Make/O/N=2 showLevel = level;CopyScales/I peaks, showLevel

   Display /W=(35.25,41,856.5,249.5) peaks
   AppendToGraph risingYs vs risingEdges
   AppendToGraph fallingYs vs fallingEdges
   AppendToGraph showLevel
   ModifyGraph mode(risingYs)=8,mode(fallingYs)=8
   ModifyGraph marker(risingYs)=17,marker(fallingYs)=23
   ModifyGraph lStyle(showLevel)=1
   ModifyGraph
   rgb(risingYs)=(19675,39321,1),rgb(fallingYs)=(0,0,0),rgb(showLevel)=(1,16019,65535)
   Legend/C/N=text0/X=3.55/Y=1.90
End
```

See Also

The **FindLevel** operation for details about the level crossing detection algorithm and the /B, /P, /Q, /R, and /T flag values.

The FindLevels operation is not multidimensional aware. See **Analysis on Multidimensional Waves** on page II-95 for details.

FindListItem

`FindListItem(`*`itemStr, listStr`*`[, `*`listSepStr`*`[, `*`start`*`[, `*`matchCase`*`]]])`

The FindListItem function returns the byte offset into *listStr* where *itemStr* begins. *listStr* should contain items separated by the *listSepStr*.

Use FindListItem to locate the start of an item in a string containing a "wave0;wave1;" style list such as those returned by functions like **TraceNameList** or **AnnotationList**, or a line from a delimited text file.

Use WhichListItem to determine the index of an item in the list.

If *itemStr* is not found, if *listStr* is `""`, or if *start* is not within the range of 0 to strlen(*listStr*)-1, then -1 is returned.

listSepStr, *startIndex*, and *matchCase* are optional; their defaults are ";", 0, and 1 respectively.

Details

ItemStr may have any length.

listStr is searched for the first instance of the item string bound by a *listSepStr* on the left and a *listSepStr* on the right. The returned number is the byte index where the first character of *itemStr* was found in *listSepStr*.

The search starts from the byte position in *listStr* specified by *start*. A value of 0 starts with the first character in *listStr*, which is the default if *start* is not specified.

listString is treated as if it ends with a *listSepStr* even if it doesn't.