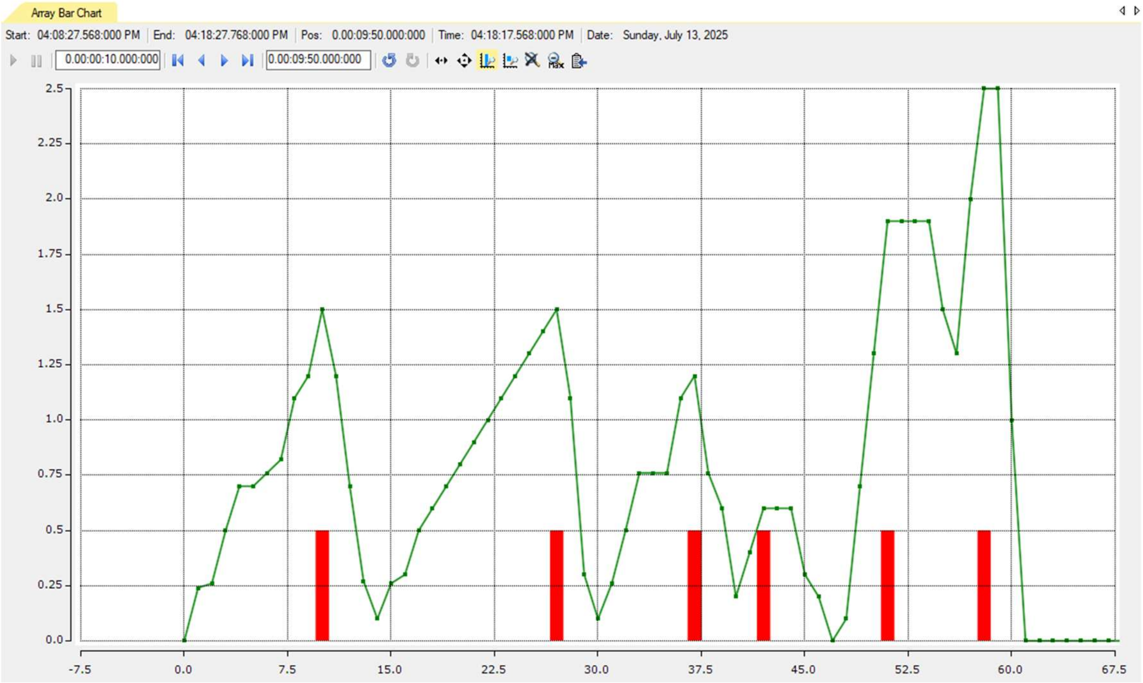
Find Peaks in 1-D Signal Array

This code takes a 1-D array and finds all local maxima by simple comparison of neighboring values.

Input parameter:

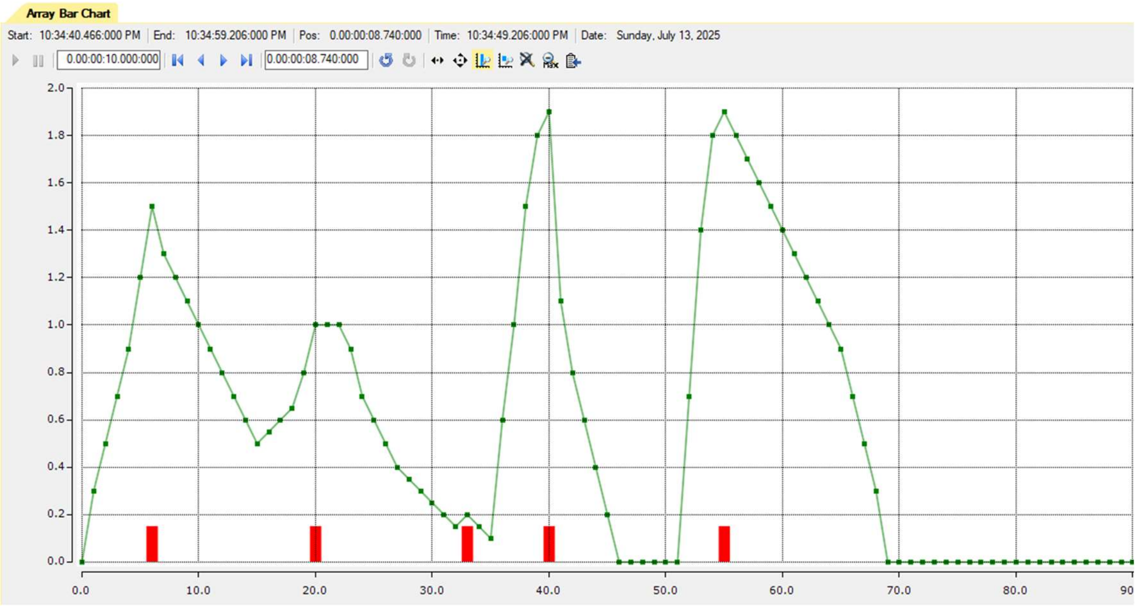
VertThresh – vertical threshold – the minimum vertical distance between neighbore points to detect peak.

PlateauMax – maximum number of points in flat top of the peak. If plateau is longer then this parameter then it is not a peak but only flat signal.



Start: 04:08:27.568:000 PM | End: 04:18:27.768:000 PM | Pos: 0.00:09:50.000:000 | Time: 04:18:17.568:000 PM | Date: Sunday, July 13, 2025

0.00:00:10.000:000 | 0.00:09:50.000:000



Start: 10:34:40.466:000 PM | End: 10:34:59.206:000 PM | Pos: 0.00:00:08.740:000 | Time: 10:34:49.206:000 PM | Date: Sunday, July 13, 2025

0.00:00:10.000:000 | 0.00:00:08.740:000

```
PROGRAM MAIN
VAR
       FindPeaks01                  : SignalFindPeaks;

       arrTestSignals01             :      ARRAY[0..99] OF REAL :=
                                    [0, 0.24, 0.26, 0.5, 0.7, 0.7, 0.76, 0.82, 1.1, 1.2, 1.5, 1.2,
                                    0.70, 0.27, 0.1,0.26, 0.3, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1,
                                    1.2, 1.3, 1.4, 1.5, 1.1, 0.3, 0.1, 0.26, 0.5, 0.76, 0.76,
                                    0.76, 1.1, 1.2, 0.76, 0.6, 0.2, 0.4, 0.6, 0.6, 0.6, 0.3, 0.2,
                                    0.0, 0.1, 0.7, 1.3, 1.9, 1.9, 1.9, 1.9, 1.5, 1.3, 2.0, 2.5,
                                    2.5, 1.0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                                    0];

       arResultPeaksIdx      : ARRAY [0..800] OF DINT;
       arResultPeaksVal      : ARRAY [0..800] OF REAL;
END_VAR


FindPeaks01(
       Enable:= TRUE,
       VertThresh:=0.005 ,
       PlateauMax:= 5,
       RawSignal:= arrTestSignals02,
       PeakSignalIdx:= arResultPeaksIdx,
       PeakSignalVal:= arResultPeaksVal,
       LastIdxRawSignal=> );
```

```
FUNCTION_BLOCK SignalFindPeaks
VAR_INPUT
       Enable                :      BOOL;         // enable FB execution

       VertThresh            :      REAL;         // minimum vertical distance between
                                                  // points to detect peaks

       PlateauMax            :      UINT;         // maximum number of points in flat top of
                                                  // a peak. If plateau is longer then this
                                                  // parameter, its not a peak but only flat signal
END_VAR
VAR CONSTANT

       SampleSize            :      DINT := 800;  // max. sample size.For 10ms task cycle = 8sec
END_VAR
VAR
       // array with one's on positions with peaks
       PeakSignalPresent     :      ARRAY[0..SampleSize] OF DINT;

       UpBound               :      DINT;         // max index of input signal array
       LowBound              :      DINT;         // min index of input signal array
       Idx                   :      DINT;         // index in FOR loop
       SubIdx                :      DINT;         // sub index in FOR loop during plateau checking
       NoPeakIdx             :      DINT;         // temporary index for peak counter
END_VAR

VAR_IN_OUT
       // input signal as array
       RawSignal                    :      ARRAY[*] OF REAL;
       // output - indexes of peaks in raw input signal
       PeakSignalIdx                :      ARRAY[0..SampleSize] OF DINT;
       // output - values of peaks
       PeakSignalVal                :      ARRAY[0..SampleSize] OF REAL;
END_VAR

VAR_OUTPUT
       LastIdxRawSignal             :      DINT;  // index of last peak in raw signal array
END_VAR
```

```
(* This FB detects peaks in incoming signal array 'RawSignal' *)

IF ENABLE THEN
    // clear array with peak signal (prepare for new evaluation)
    MemSet(ADR(PeakSignalPresent),0,SIZEOF(UDINT) * (SampleSize));

    // clear array with peak indexes in RawSignal array
    MemSet(ADR(PeakSignalIdx),0,SIZEOF(UDINT) * (SampleSize));
    // clear array with peak values from RawSignal array
    MemSet(ADR(PeakSignalVal),0,SIZEOF(UDINT) * (SampleSize));

    UpBound         := UPPER_BOUND(RawSignal,1);  // find max index of input signal array
    LowBound        := LOWER_BOUND(RawSignal,1);  // find min index of input signal array
    NoPeakIdx       := 0;
    LastIdxRawSignal          := 0;

    IF LowBound < UpBound AND UpBound-LowBound > 5 THEN
          // iterate through all points in raw signal array
          FOR Idx := LowBound+1 TO UpBound-1 BY 1 DO
              // #1 if predecessor and successor of current signal point are less then signal point -> It's peak point
                  IF (RawSignal[Idx] > RawSignal[Idx-1] + VertThresh) AND (RawSignal[Idx] > RawSignal[Idx+1] + VertThresh) THEN
                              PeakSignalPresent[Idx] := 1;
                              PeakSignalIdx[NoPeakIdx] := Idx;
                              PeakSignalVal[NoPeakIdx] := RawSignal[Idx];
                              NoPeakIdx:=NoPeakIdx+1;

              // #2 predecessor is less and successor is equal to current signal pont
                  ELSIF (RawSignal[Idx]  > RawSignal[Idx-1] + VertThresh) AND (ABS(RawSignal[Idx] - RawSignal[Idx+1]) <0.001) THEN
                      // iterate right side area from current signal point
                      FOR SubIdx:= Idx+1 TO UpBound BY 1 DO
                          IF RawSignal[SubIdx] - RawSignal[Idx] > 0 THEN
                              // stop searching and exit when successor point is higher then current signal point. It can't be peak
                              EXIT;
                          // if successor is less then current point then its peak.
                          ELSIF (RawSignal[Idx] - RawSignal[SubIdx] > VertThresh) AND (SubIdx-Idx < PlateauMax) THEN
                              PeakSignalPresent[Idx] := 1;
                              PeakSignalIdx[NoPeakIdx] := Idx;
                              PeakSignalVal[NoPeakIdx] := RawSignal[Idx];
                              NoPeakIdx:=NoPeakIdx+1;
                              EXIT;
                          // if number of following equal points higher then plateau_max then stop searching. Its not peak but flat area signal
                          ELSIF SubIdx-Idx >  PlateauMax THEN
                              EXIT;
                          END_IF;
                      END_FOR
                  END_IF;
          END_FOR


          // indicate how many peaks detected
          IF NoPeakIdx > 0 THEN
              LastIdxRawSignal          :=        NoPeakIdx;
          ELSE
              LastIdxRawSignal          := 0;
          END_IF
    END_IF

          MemSet(ADR(RawSignal),0,SIZEOF(RawSignal));
END_IF;
```