

DeMarker Study

The DeMarker study by Tom Demark is an attempt to overcome the shortcomings of classical overbought / oversold indicators. The DeMarker study identifies potential price bottoms and tops. It accomplishes this by making price comparisons from one bar to the next and measuring the level of price demand.

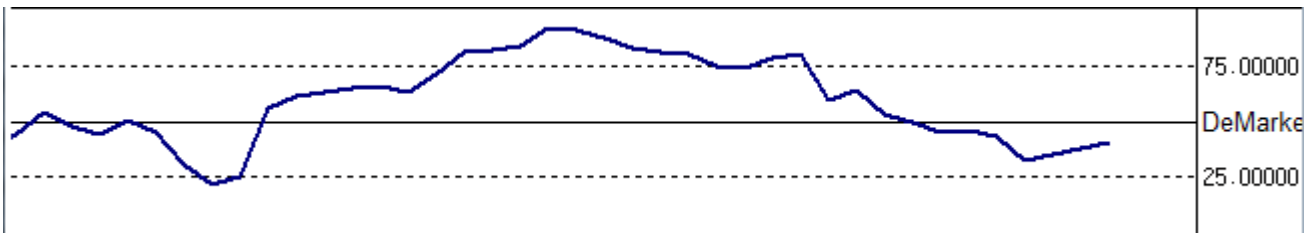
Formula

highdelta = IIF(High > PriorHigh, High - PriorHigh, 0);

lowdelta = IIF(Low < PriorLow, PriorLow - Low, 0);

DeMarker = 100 * Sum(highdelta, 13) / (Sum(lowdelta, 13) + Sum(highdelta, 13));

Category	Variable	Selection #1 & #3	Op. [#]	Selection #2 & #4	Offset	Show Marker	Color
Action	22 HighSpread	= if ## then V := (#3 - #4) els		0	0	<input type="checkbox"/>	
		(High	>	Prior High	0	<input type="checkbox"/>	
A	Action	if (High > Prior High) then [HighSpread] := (High - Prior High) else		<input type="checkbox"/>			
B	Action	if (Prior Low > Low) then [LowSpread] := (Prior Low - Low) else		<input type="checkbox"/>			
C	Function	[SumHigh] := Sum([A], 13)		<input type="checkbox"/>			
D	Function	[SumLow] := Sum([B], 13)		<input type="checkbox"/>			
E	Expression	[Demarker] := (100 * [SumHigh]) / ([SumHigh] + [SumLow])		<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	Study Value

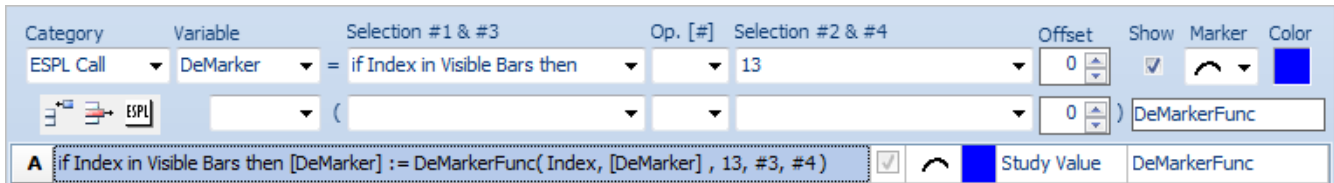


C – The Sum function references the values on row A, and not the variable HighDelta. The variable has a single value, whereas, row A has a series of values that can be summed.

D – See the comment for row C. The Sum function needs to reference row B values.

ESPL Script

For the sake of comparison and illustration, the DeMarker study will be implemented with an ESPL script.



A – The function call will be made just for the visible bars to reduce the CPU calculation burden. The parameter for the Sum function will be passed in the Selection #2 field.

This ESPL script implements the DeMarker study.

```
ESPL Editor
6 function DeMarkerFunc(index, v, count, n3, n4);
7 begin
8   SumHigh := 0;
9   SumLow  := 0;
10  for j := index - count + 1 to index do begin
11    if High(j) > High(j-1) then SumHigh := SumHigh + High(j) - High(j-1);
12    if Low(j) < Low(j-1) then SumLow := SumLow + Low(j-1) - Low(j);
13  end;
14  if SumHigh+SumLow > 0 then Result := 100 * SumHigh / (SumHigh + SumLow)
15  else Result := 0;
16 end;
```

Index – This parameter is the bar index passed by the DYO for the bar being evaluated.

Count – This is the parameter passed by the DYO from the Selection #2 field, which is a 13.

High(index) - The High array holds the bar high values. (see line 11)

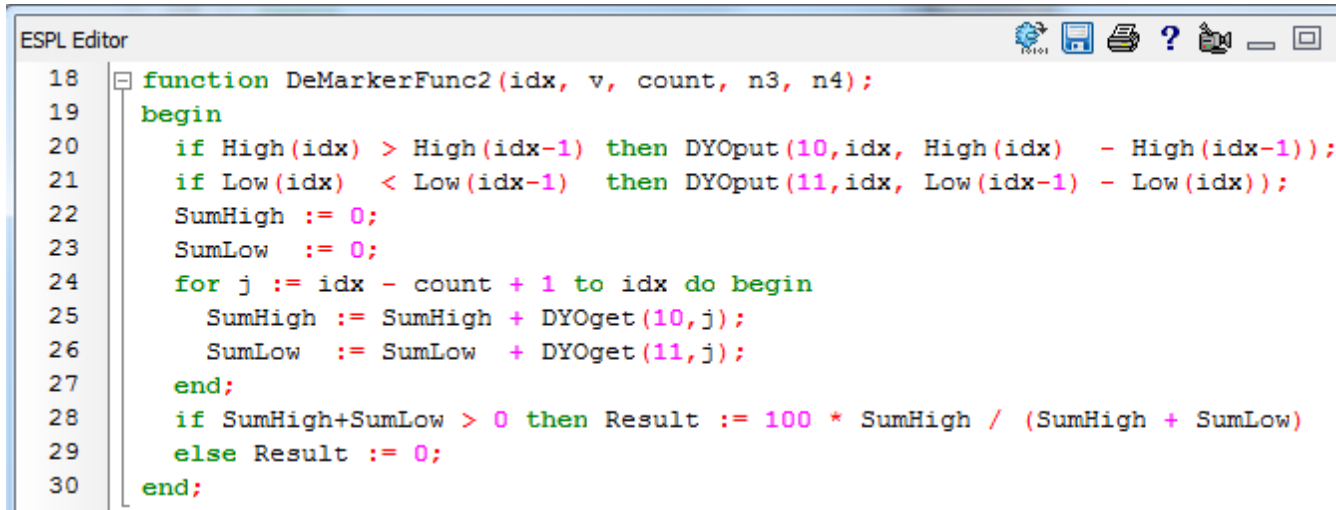
Low(index) - The Low array holds the bar low values. (see line 12)

Speed – The ESPL implementation, unfortunately, requires more CPU cycles to calculate. The DYO calculation time for 1650 bars was 0.01 seconds, whereas the ESPL calculation time for the same 1650 bars was 0.43 seconds. Four tenths of a second might not be noticed when opening a chart containing 1650 bars, but would be noticed if the ESPL calculation was performed on 20,000 bars.

Wisely choose an ESPL Call selection that will minimize the calculation burden. The example shown calculates for just the visible bars, which for a typical chart might be 65 bars though the chart file might contain thousands of bars. The ESPL calculation time for 65 visible bars was 0.03 seconds.

DYO Arrays

This variation of the DeMarker script is more efficient by doing the bar calculations one time per call, and saving the calculations to DYO row arrays. (lines 20-21) The loop for the sums reads the values stored in the DYO arrays. (lines 24-27). The calculation time across 1650 bars dropped from 0.43 to 0.23 seconds.



```
ESPL Editor
18 function DeMarkerFunc2(idx, v, count, n3, n4);
19 begin
20     if High(idx) > High(idx-1) then DYOpout(10,idx, High(idx) - High(idx-1));
21     if Low(idx) < Low(idx-1) then DYOpout(11,idx, Low(idx-1) - Low(idx));
22     SumHigh := 0;
23     SumLow := 0;
24     for j := idx - count + 1 to idx do begin
25         SumHigh := SumHigh + DYOget(10,j);
26         SumLow := SumLow + DYOget(11,j);
27     end;
28     if SumHigh+SumLow > 0 then Result := 100 * SumHigh / (SumHigh + SumLow)
29     else Result := 0;
30 end;
```

DYOput(row, index, value) - This function writes the Value parameter into one of the DYO row arrays, 1 through 12, at the bar's index position. Row A is array 1, and row L is array 12. Since the example DYO only used row A, the other 11 arrays were available for general storage.


DYOget(row, index) - This function returns a Value from one of the DYO row arrays, 1 through 12.


It is a good programming practice to verify that the denominator is non-zero to avoid getting the error 'Invalid Float Operation' if a division by zero were to be performed. (lines 28-29)

A DYO design can be a combination of DYO statements and calls to ESPL functions. As demonstrated in this last example, ESPL has full access to read and write DYO row arrays. For example, the ESPL function could have written the row values used by the DYO function to Sum(#2, [#]), thus avoiding a time consuming loop in the ESPL function.

ESPL File

A single ESPL script file is used by all DYO's. This ESPL script may contain multiple functions.

 The file path and name is C:\Ensign10\ESPL\DYO.psc. The file is automatically loaded for use by any DYO and is edited by clicking the ESPL button on the DYO property form. Edits are saved to this file.

 The first button on the Editor toolbar is the Compile button. Click this button to verify the ESPL script syntax is correct and it will compile.