Use of VB to Enhance Standard BEx Features – A Case Study

Applies to:

Development and support based on SAP BI 7.0 or SAP BW 3.5.

Summary

This article focuses on when it might be viable to use VB in BEx workbooks to build on standard BEx features. Different ways of incorporating VB in workbooks, as well as pros and cons of using VB with standard SAP BI are covered

Author: Amrita Goswami

Company: Infosys Technologies Ltd

Created on: 16 June 2009

Author Bio

Author is a BI Consultant having worked in all phases of Project Implementation in SAP BI/BW in a global delivery environment. She also has experience in SAP Variant Configuration and in the Manufacturing Industry.

Table of Contents

Why do we need VB in BI Workbooks?

BEx Query Designer provides the BI Developer a comprehensive framework to design the Report layout. However, it does not support features like calculations on the resultant data set after query run time. Also the Key Figure Totals can have some standard evaluations such as : Summation, Average, Maximum, Minimum, Counter, Standard Deviation etc. Any custom calculations on the Totals are not supported. Virtual Key Figures can take care of some of these limitations however they are added into the query as additional columns and may not always serve customer needs.

To overcome this SAP BI Workbooks with VBA add-in can be used. This enables us to leverage the features of Excel and VB and to provide a highly user friendly and interactive Report output.

Standard VBA Routines provided by SAP

SAP provides a BEx add-in provided with several standard VBA Routines which can be used to enhance report layout or calculations. Some of the functions available in SAP BI Version 7.0 are listed below:

Function Name	Description
SAPBEXgetWorkbookID(wbName As String) As String	Determining the ID, under which a workbook is saved in the InfoCatalog
SAPBEXreadWorkbook(wbID As String) As String	Reading a workbook from the InfoCatalog
SAPBEXsaveWorkbook(wbName As String) As Integer	Saving a workbook in the InfoCatalog
SAPBEXsetFilterValue(intValue As String, Optional hierValue As String, Optional atCell As Range) As Integer	Filtering with cell atCell (if this parameter is missing, then the active cell is taken as the basis)
SAPBEXgetFilterValue(intValue As String, hierValue As String, Optional atCell As Range) As Integer	Determining the internal filter value at Cell
SAPBExcopyFilterValue Function (fromCell As Range, Optional atCell As Range) As Integer	Copying a Filter Value from Filter Cells fromCell to Filter Cells atCell
SAPBEXsetDrillState(newState As Integer, Optional atCell As Range) As Integer	Change drilldown status with cell atCell
SAPBEXgetDrillState(currentState As Integer, Optional atCell As Range) As Integer	Determining the drilldown status with cell atCell
SAPBEXrefresh(allQueries As Boolean, Optional atCell As Range) As Integer	allQueries = FALSE: Refreshing the query with cell atCell allQueries = TRUE: Refreshing all queries in the active workbook
SAPBEXpauseOn() SAPBEXpauseOff()	Stopping & Starting the process run in the add-in
SAPBEXgetFilterValue(intValue As String, hierValue As String, Optional atCell As Range) As Integer	Jumps to a view that is defined in the workbook.
SAPBEXembedQuery (genUID As String, Optional atActiveCell As Boolean) As String	Embedding a query that is given by generated unique ID (genUID)

For example one can use the following sample macro in BI Workbooks:

```
Sub SAPBEXonRefresh(queryID As String, resultArea As Range)

** Start of custom code to remove #
Dim c As Range
For Each c In resultArea.Cells
If c.Value = "#" Then c.Value = ""
Next c

** End of custom code to remove #
End Sub
```

Here using standard BEx add-ins hash (#) is removed from the resultant data. This Code executes every time the Workbook is refreshed.

Scenarios where VB can be used

- User wants to see calculated values using the data from two reports.
- User wants to display extended text in the workbook.
- User wants special formatting in the Report like merging cells, making text bold, highlighting columns or cells.

When to use VB in BI Workbooks- A Business Case

Customer requirement is to have a Monthly Report about Purchase, Sales and Inventory. Each Material has different Costs maintained in ECC. The cost data is considered to be valid across all months and is not assigned to any monthly bucket.

Fig 1: Report layout required

				SEPT	OCTOBER			NOVEMBER			DECEMBER		
Product Type	Model	Cost Category	Cost in $	INVEN	SALES	PURCH	INVEN	SALES	PURCH	INVEN	SALES	PURCH	INVEN
Accessory	ABC	Landed Cost	20	50	250	200	0	300	300	0	275	275	0
		FOB Cost	15										
		Sales Price	25										
Main Unit	XYZ	Landed Cost	24	100	300	200	0	400	400	0	456	456	0
		FOB Cost	20										
		Sales Price	35										
	TOTAL (Main Unit)				550	400		700	700		731	731	
	TOTAL ($$$) - Landed Cost				12200	8800							
	TOTAL ($$$) - FOB Cost				9750	7000							
	TOTAL ($$$) - Sales Price				16750	12000							
	TOTAL (ACCESSORY)												
	TOTAL ($$$) - Landed Cost												
	TOTAL ($$$) - FOB Cost												
	TOTAL ($$$) - Sales Price												
	TOTAL (ARBTR)												
	TOTAL ($$$) - Landed Cost												
	TOTAL ($$$) - FOB Cost												
	TOTAL ($$$) - Sales Price												
	TOTAL (Options)												

PSI

Note: The Totals for Sales are calculated by multiplying each Cost type (Landed, FOB, Sales Price) with the Sales Key Figure and then summing up for the entire Product Type. The Same holds for Totals Purchase and Totals Inventory.

Solution Options

1. Create a Calculated Key Figure for each of the Totals: Since Cost is not assigned to any month multiplying cost with Sales or Purchase will not give the desired results as for each Calmonth one of the factors will always be Null.

   For example:

   Calmonth = Oct, Purchase for Material ABC = 200 and Landed Cost KF = Null.

   Calmonth = # (Not Assigned) Purchase for Material ABC = Null and Landed Cost KF = 20

2. Using two queries linked with RRI: The First Query includes only the monthly values. The second query contains the Totals. In this design calculation of the Totals has to be done on the back end in the cubes itself. An RRI will be used to link the two queries. However, this may not accepted by business since this would mean swapping between 2 reports to link the PSI figures with the financial totals projected in the 2nd report.

3. Translating Costs to Monthly Buckets and creating KFs as in option 1: In this case there would be 9 additional columns for each Calmonth. Landed Cost, FOB, Sales Price, Total Landed Cost- Sales, Total Landed Cost-Purchase, Total FOB Cost- Sales, Total FOB Cost-Purchase, Total Sales Price-Sales, Total Sales Price –Purchase. This is a huge deviation from the format requested by the customer.

4. So in order to obtain a closest fit to customer requirement a VB solution using Macros in the workbook was recommended. Here Totals are calculated after Query Run Time using VBA Code.

How VB is incorporated with BEx

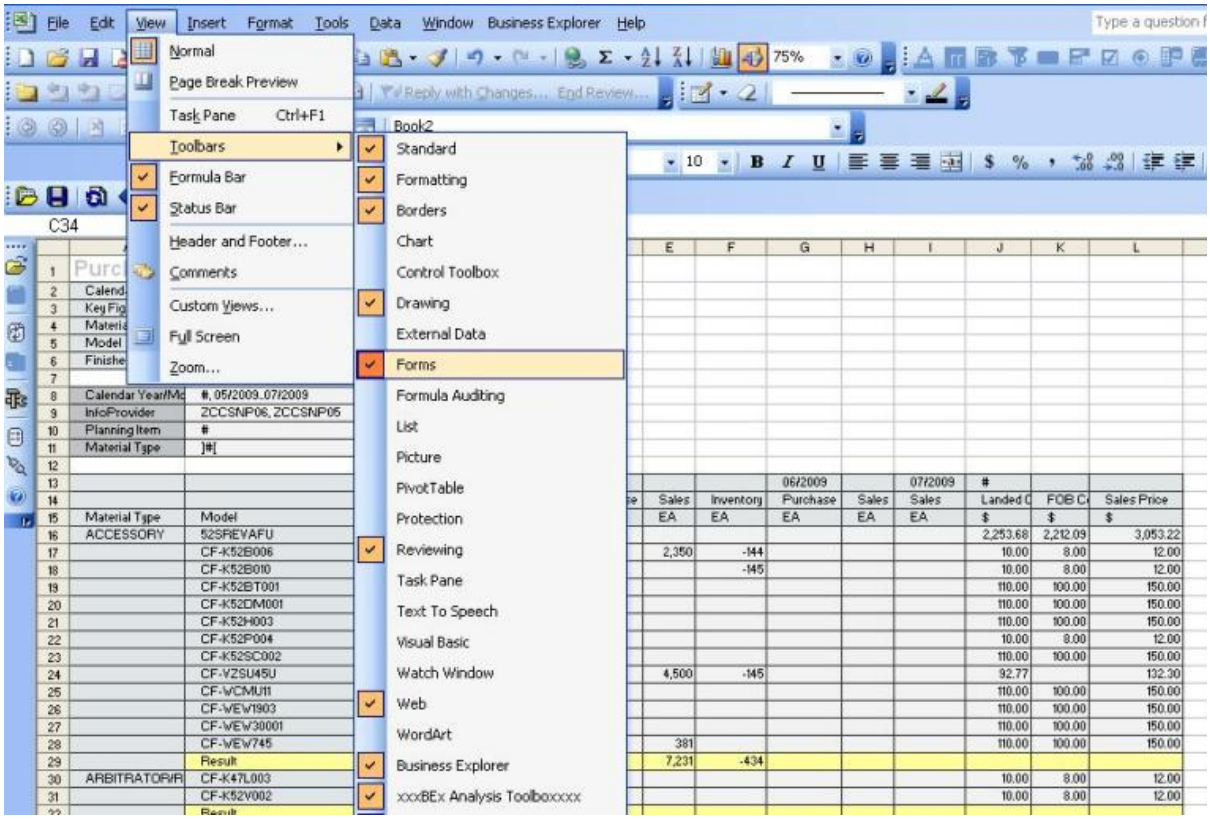
Report is generated in Query Designer with the setting ‘Always Display Totals’ Settings for the Characteristics Model. Query Output is as follows:

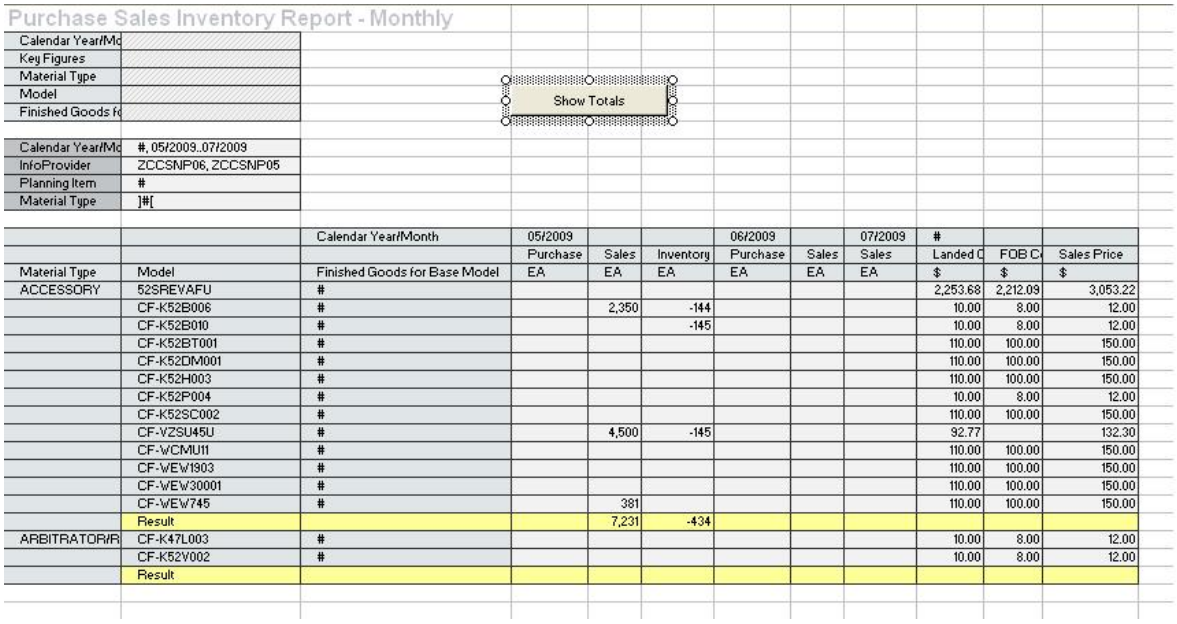
Purchase Sales Inventory Report - Monthly

Calendar Year/Mo	
Key Figures	
Material Type	
Model	
Finished Goods fo	
Calendar Year/Mo	#, 05/2009..07/2009
InfoProvider	ZCCSNP06, ZCCSNP05
Planning Item	#
Material Type	]#[

		Calendar Year/Month	05/2009			06/2009		07/2009	#		
			Purchase	Sales	Inventory	Purchase	Sales	Sales	Landed C	FOB C	Sales Price
Material Type	Model	Finished Goods for Base Model	EA	EA	EA	EA	EA	EA	$	$	$
ACCESSORY	52SREVAFU	#							2,253.68	2,212.09	3,053.22
	CF-K52B006	#		2,350	-144				10.00	8.00	12.00
	CF-K52B010	#			-145				10.00	8.00	12.00
	CF-K52BT001	#							110.00	100.00	150.00
	CF-K52DM001	#							110.00	100.00	150.00
	CF-K52H003	#							110.00	100.00	150.00
	CF-K52P004	#							10.00	8.00	12.00
	CF-K52SC002	#							110.00	100.00	150.00
	CF-VZSU45U	#		4,500	-145				92.77		132.30
	CF-WCMU11	#							110.00	100.00	150.00
	CF-WEW1903	#							110.00	100.00	150.00
	CF-WEW30001	#							110.00	100.00	150.00
	CF-WEW745	#		381					110.00	100.00	150.00
	Result			7,231	-434						
ARBITRATOR/R	CF-K47L003	#							10.00	8.00	12.00
	CF-K52V002	#							10.00	8.00	12.00
	Result										

From the Forms Toolbar add a Button called ‘Show Totals’ to the Report Output Sheet as shown below:



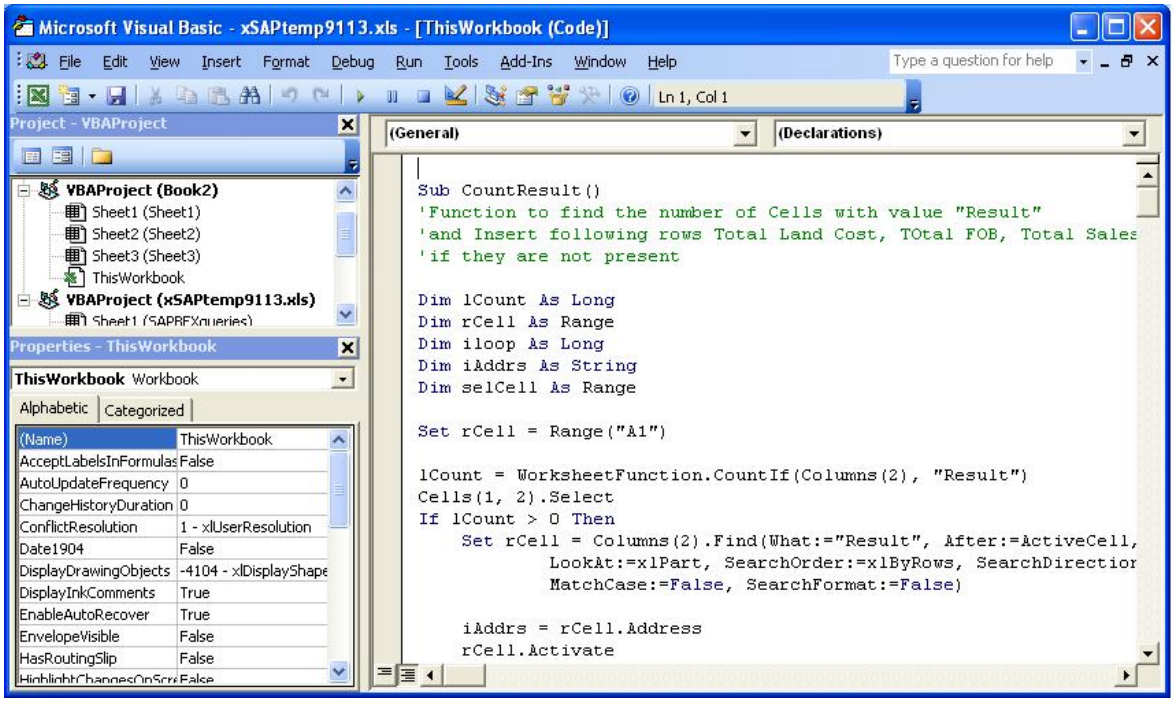


Calendar Year/Mo	
Key Figures	
Material Type	
Model	
Finished Goods fo	

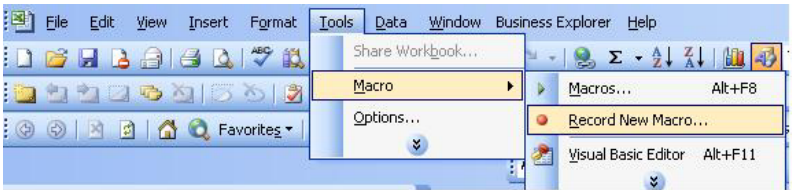
Calendar Year/Mo	#, 05/2009..07/2009
InfoProvider	ZCCSNP06, ZCCSNP05
Planning Item	#
Material Type	]#[

		Calendar Year/Month	05/2009			06/2009		07/2009	#		
			Purchase	Sales	Inventory	Purchase	Sales	Sales	Landed C	FOB C	Sales Price
Material Type	Model	Finished Goods for Base Model	EA	EA	EA	EA	EA	EA	$	$	$
ACCESSORY	52SREVAFU	#							2,253.68	2,212.09	3,053.22
	CF-K52B006	#		2,350	-144				10.00	8.00	12.00
	CF-K52B010	#			-145				10.00	8.00	12.00
	CF-K52BT001	#							110.00	100.00	150.00
	CF-K52DM001	#							110.00	100.00	150.00
	CF-K52H003	#							110.00	100.00	150.00
	CF-K52P004	#							10.00	8.00	12.00
	CF-K52SC002	#							110.00	100.00	150.00
	CF-VZSU45U	#		4,500	-145				92.77		132.30
	CF-WCMU11	#							110.00	100.00	150.00
	CF-WEW1903	#							110.00	100.00	150.00
	CF-WEW30001	#							110.00	100.00	150.00
	CF-WEW745	#		381					110.00	100.00	150.00
	Result			7,231	-434						
ARBITRATOR/R	CF-K47L003	#							10.00	8.00	12.00
	CF-K52V002	#							10.00	8.00	12.00
	Result										

Now Code is written in a Macro in the Visual Basic Editor and Macro is assigned to the Show Totals Button seen above:



Note: For simple macros the Macro Recorder can be used to convert into VBA code all actions done on the Excel file in the frontend. This is especially helpful for new users who are not familiar with VB syntax and standards.



Logic written in VB Macro

Whenever Result appears in column B insert three rows below 'Result' row.

In the inserted rows in Column B write Total Landed Cost, Total FOB Cost, and Total Sales Price.

Row Total Landed Cost:

Wherever Purchase occurs as a heading in the columns in that column formula to be used is ∑ (Purchase * Landed Cost)

Wherever Sales occurs as a heading in the columns in that column formula to be used is ∑ (Sales * Landed Cost)

Row Total FOB Cost:

Wherever Purchase occurs as a heading in the columns in that column formula to be used is ∑ (Purchase * FOB Cost)

Wherever Sales occurs as a heading in the columns in that column formula to be used is ∑ (Sales * FOB Cost)

Row Total Sales Price:

Wherever Purchase occurs as a heading in the columns in that column formula to be used is ∑ (Purchase * Sales Price)

Wherever Sales occurs as a heading in the columns in that column formula to be used is ∑ (Sales * Sales Price)

This is to be done every time 'Result' appears in the Column B and for every Calmonth appearing in the Report, in other words for every Material Type the three Totals will be calculated.

Workbook Output on Executing the Macro:

Purchase Sales Inventory Report - Monthly

Calendar Year/Mc	
Key Figures	
Material Type	

Show Totals

Calendar Year/Mc	.05/2009..07/2009
InfoProvider	
Planning Item	
Material Type	

		Calendar Year/Month	05/2009			06/2009		07/2009			
			Purchase	Sales	Inventory	Purchase	Sales	Sales	Landed C	FOB C	Sales Price
Material Type	Model	Finished Goods for Base Model	EA	EA	EA	EA	EA	EA	$	$	$
ACCESSORY	52SREVAFU								2,253.68	2,212.09	3,053.22
	CF-K52B006			2,350	-144				10.00	8.00	12.00
	CF-K52B010				-145				10.00	8.00	12.00
	CF-K52BT001								110.00	100.00	150.00
	CF-K52DM001								110.00	100.00	150.00
	CF-K52H003								110.00	100.00	150.00
	CF-K52P004								10.00	8.00	12.00
	CF-K52SC002								110.00	100.00	150.00
	CF-VZSU45U			4,500	-145				92.77		132.30
	CF-WCMU11								110.00	100.00	150.00
	CF-WEW1903								110.00	100.00	150.00
	CF-WEW30001								110.00	100.00	150.00
	CF-WEW745			381					110.00	100.00	150.00
	Total			7,231	-434						
	Total Landed Cost		0.00	482,875.00	-16,341.65	0.00	0.00	0.00			
	Total FOB		0.00	56,900.00	-2,312.00	0.00	0.00	0.00			
	Total Sales Price		0.00	680,700.00	-22,651.50	0.00	0.00	0.00			
ARBITRATOR/R	CF-K47L003								10.00	8.00	12.00
	CF-K52V002								10.00	8.00	12.00
	Total										
	Total Landed Cost		0.00	0.00	0.00	0.00	0.00	0.00			

Here we see three new rows have been added to the Query Output: Total Landed Cost, Total FOB Cost & Total Sales Price

Also the values in these rows have been calculated using the logic given above. Some cosmetic changes are also done in the output.

Thus by using VB in Workbook we are able to meet the customer requirement of seeing Key Figures as well as Totals for each Category in the same sheet at the press of a button.

Instead of using a button, the function SAPBEXonRefresh can also be used to run the macro. The choice can be made depending on user preferences.

Pros and Cons of Using VB in BEx Workbooks:

Pros

- The use of Visual Basic Macros in BI workbooks enables us to leverage the features of Excel based Reporting through Workbooks.
- It provides greater flexibility in designing the layout and formatting of reports.
- It enables us to perform operations on the resultant data set after query runtime that standard BEx does not support.
- For the greenhorn SAP BI user, Excel based reporting using workbooks and VB ensures greater ease of operation.

Cons

- During Upgrade from one version of SAP BI to the next, custom VB code in the Workbooks may not be automatically migrated correctly. This has to be done manually and tested to ensure all features are working as desired.
- Visual Basic is not a core component of SAP BI consulting and this might lead to issues in Supporting Projects using extensive VB coding.
- In MS Excel the Security Level must be set to low in client machines to ensure the macros embedded in Workbook can be run.

Sample Code in VB for achieving this functionality:

```
Sub CountResult()
'Function to find the number of Cells with value "Result"
'and Insert following rows Total Landed Cost, TOtal FOB, Total Sales Price
'if they are not present
Dim lCount As Long
Dim rCell As Range
Dim iloop As Long
Dim iAddrs As String
Dim selCell As Range
Set rCell = Range("A1")
lCount = WorksheetFunction.CountIf(Columns(2), "Result")
Cells(1, 2).Select
If lCount > 0 Then
   Set rCell = Columns(2).Find(What:="Result", After:=ActiveCell, LookIn:=xlFormulas, _
         LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
         MatchCase:=False, SearchFormat:=False)

   iAddrs = rCell.Address
   rCell.Activate
   For iloop = 1 To lCount
   Call InsROWS(iAddrs)  ' Calling Insert rows Function
      Set rCell = Columns(2).FindNext(After:=ActiveCell)
      rCell.Activate
      iAddrs = rCell.Address
   Next iloop
End If
End Sub
Sub InsROWS(iAddrs As String)
'Function to Insert the 3 rows below the Result cell found in Column B
If Not Range(iAddrs).Offset(1, 0) = "Total Landed Cost" Then
   Range(iAddrs).Offset(1, 0).Activate
   Range(iAddrs).Offset(1, 0).EntireRow.Insert
   Selection.Value = "Total Landed Cost"
   ActiveCell.EntireRow.Select
   Selection.NumberFormat = "#,##0.00"
End If
If Not Range(iAddrs).Offset(2, 0) = "Total FOB" Then
```

```
    Range(iAddrs).Offset(2, 0).Activate
    Range(iAddrs).Offset(2, 0).EntireRow.Insert
    Selection.Value = "Total FOB"
    ActiveCell.EntireRow.Select
    Selection.NumberFormat = "#,##0.00"
End If

If Not Range(iAddrs).Offset(3, 0) = "Total Sales Price" Then
    Range(iAddrs).Offset(3, 0).Activate
    Range(iAddrs).Offset(3, 0).EntireRow.Insert
    Selection.Value = "Total Sales Price"
    ActiveCell.EntireRow.Select
    Selection.NumberFormat = "#,##0.00"
    Range(iAddrs).Offset(3, 0).Select
End If
End Sub

Sub CalcResult()
'Function to calc the Values in Rows Total Landed Cost, TOtal FOB, Total Sales Price
Dim rCell As Range
Dim colValue As Long
Dim RowHdVal As Long
Dim AdrsLanded As String
Dim AdrsFOB As String
Dim AdrsSalesP As String
Dim PurLstCell As Double
Dim PurFOB As Double
Dim PurSP As Double
Dim AdrsPur As String
Dim SalLand As Double
Dim SalFOB As Double
Dim SalSP As Double
Dim InvLand As Double
Dim InvFOB As Double
Dim InvSP As Double
Dim AdrsInv As String
Dim AdrsSal As String
Dim n As Long
Dim lCount As Long
```

```
Dim rNext As Long
Dim rPrevious As Long
Dim AdrsEnd As String

lCount = WorksheetFunction.CountIf(Columns(2), "Result")
rLstCell = Columns(2).SpecialCells(xlLastCell).Row 'Last cell in column B
colValue = (Rows(13).SpecialCells(xlLastCell).Column) 'Last Cell in Header Row
'Row where the columns headers are specified (Header Row)
For rLoop = 1 To rLstCell
If Cells(rLoop, 2).Value = "Model" Then
RowHdVal = rLoop - 2
Exit For
End If
Next rLoop

rNext = RowHdVal
For iloop = 1 To lCount
rPrevious = rNext + 1
Rows(RowHdVal + 1).Select
Selection.Find(What:="Landed Cost", After:=ActiveCell, LookIn:=xlFormulas, _
LookAt:=xlPart, SearchOrder:=xlByRows, SearchDirection:=xlNext, _
MatchCase:=False, SearchFormat:=False).Activate
‘Find the address of cells where value is Total Landed Cost, Total FOB Cost, Total Sales Price
For jLoop = 1 To colValue
If Cells(RowHdVal + 1, jLoop).Value = "Landed Cost" Then
   AdrsLanded = Cells(rNext, jLoop).Address
End If
If Cells(RowHdVal + 1, jLoop).Value = "FOB Cost" Then
   AdrsFOB = Cells(rNext, jLoop).Address
End If
If Cells(RowHdVal + 1, jLoop).Value = "Sales Price" Then
   AdrsSalesP = Cells(rNext, jLoop).Address
End If
Next jLoop
For kLoop = 1 To colValue
'Loop through the header row till the last empty Cell in that row
PurLstCell = 0
PurFOB = 0
PurSP = 0
```

```
SalLand = 0
SalFOB = 0
SalSP = 0
InvLand = 0
InvFOB = 0
InvSP = 0
‘Calculate Totals in Inventory Column
If Cells(RowHdVal + 1, kLoop).Value = "Inventory" Then
AdrsInv = Cells(rNext, kLoop).Address
n = 1
For p = rNext To rLstCell

If Cells(p, 2).Value = "Result" Then
   Exit For
Else
   InvLand = InvLand + (Range(AdrsInv).Offset(n + 2, 0) * Range(AdrsLanded).Offset(n + 2, 0))
   InvFOB = InvFOB + (Range(AdrsInv).Offset(n + 2, 0) * Range(AdrsFOB).Offset(n + 2, 0))
   InvSP = InvSP + (Range(AdrsInv).Offset(n + 2, 0) * Range(AdrsSalesP).Offset(n + 2, 0))
   n = n + 1
End If
Next p
'Enter the Calc values into their respective cells
Range(AdrsInv).Offset(n, 0).Select
ActiveCell.Value = InvLand
Range(AdrsInv).Offset(n + 1, 0).Select
ActiveCell.Value = InvFOB
Range(AdrsInv).Offset(n + 2, 0).Select
ActiveCell.Value = InvSP
End If
'End Calculation of Totals in Inventory Column
'Start Calculation of Totals in Purchase Column
If Cells(RowHdVal + 1, kLoop).Value = "Purchase" Then
AdrsPur = Cells(rNext, kLoop).Address
n = 1
For p = rNext To rLstCell
If Cells(p, 2).Value = "Result" Then
   Exit For
Else
   PurLstCell = PurLstCell + (Range(AdrsPur).Offset(n + 2, 0) * Range(AdrsLanded).Offset(n + 2, 0))
```

```
    PurFOB = PurFOB + (Range(AdrsPur).Offset(n + 2, 0) * Range(AdrsFOB).Offset(n + 2, 0))
    PurSP = PurSP + (Range(AdrsPur).Offset(n + 2, 0) * Range(AdrsSalesP).Offset(n + 2, 0))
    n = n + 1
End If
Next p

'Enter the Calc values into their respective cells
Range(AdrsPur).Offset(n, 0).Select
ActiveCell.Value = PurLstCell
Range(AdrsPur).Offset(n + 1, 0).Select
ActiveCell.Value = PurFOB
Range(AdrsPur).Offset(n + 2, 0).Select
ActiveCell.Value = PurSP
End If
'End Calculation of Totals in Purchase Column

'Start Calculation of Totals in Sales column
If Cells(RowHdVal + 1, kLoop).Value = "Sales" Then
AdrsSal = Cells(rNext, kLoop).Address

n = 1
For p = rNext To rLstCell
If Cells(p, 2).Value = "Result" Then
    p = rLstCell
Else
    SalLand = SalLand + (Range(AdrsSal).Offset(n + 2, 0) * Range(AdrsLanded).Offset(n + 2, 0))
    SalFOB = SalFOB + (Range(AdrsSal).Offset(n + 2, 0) * Range(AdrsFOB).Offset(n + 2, 0))
    SalSP = SalSP + (Range(AdrsSal).Offset(n + 2, 0) * Range(AdrsSalesP).Offset(n + 2, 0))
    n = n + 1
End If
Next p
'Enter the Calc values into their respective cells
Range(AdrsSal).Offset(n, 0).Select
ActiveCell.Value = SalLand
Range(AdrsSal).Offset(n + 1, 0).Select
ActiveCell.Value = SalFOB
Range(AdrsSal).Offset(n + 2, 0).Select
ActiveCell.Value = SalSP
End If
```

```
'End Calculation of Totals in Sales column
Next kLoop
For aLoop = rPrevious To rLstCell
If Cells(aLoop, 2).Value = "Total Landed Cost" Then
rNext = aLoop
Exit For
End If
Next aLoop
Next iloop
End Sub
Sub Totals()
CountResult
CalcResult
End Sub
```

Related Content

http://help.sap.com/saphelp_nw04/helpdata/en/f1/0a55f9e09411d2acb90000e829fbfe/frameset.htm

https://www.sdn.sap.com/irj/scn/go/portal/prtroot/docs/library/uuid/70a6fd1b-21c9-2b10-ba9b-ed7660b8a579

https://www.sdn.sap.com/irj/scn/thread?messageID=1719655

For more information, visit the Business Intelligence homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.