Ken's Excel 2007 VBA Notes

Contents

Excel Cheat Sheet ... 3
Null Comparison - Empty Cells Tricks ... 3
VLOOKUP ... 3
Formula VLOOKUP ... 4
Programmatic VLOOKUP ... 4
Match ... 4
Correcting Match for the Real Row ... 4
Iterating through all the Worksheets ... 4
SumIfs ... 5
Sumifs Formula Bar Version ... 5
Sumifs Code Version ... 5
CountIf ... 5
AverageIfs ... 5
DateDiff ... 6
Combining (And, Or) Logic ... 6
Clearing Cells ... 6
Cell Style ... 6
Named Styles ... 6
Bold ... 6
Font Foregrount ... 7
Font Background ... 7
Default ColorIndex Pallet ... 7
Misc Styling ... 7
Inserting Rows ... 7
Deleting a Row ... 10
Delete a row if an entire column is blank ... 10
Delete a row if the entire row is blank ... 10
Copying a row between Worksheets ... 10
Determining Worksheet Size ... 11
Rows ... 11
Columns ... 11
Copying just the Visible (filtered) Cells ... 11
Copying between Workbooks ... 12
Deleting worksheets ... 12
Deleting All Worksheets of a Type ... 12
Deleting Worksheets that Might Not Exist ... 12
Does Worksheet Exist ... 13
Method 1 ... 13
Method 2 ... 13
Date Time ... 14
Common Block of Date Code ... 15
Time ... 16
Convert datetime column to just display the date ... 16
Weekday ... 16
Range ... 16
Column Number to Letter ... 17
Subroutines & Functions ... 18
Optional Subroutine Parm ... 18
System Calls & Time Delay ... 18
Arrays ... 19

Split 19
Splitting off filename from path 19
Getting last array element 19
Collections 20
Turning a Column into a Collection 20
Error Handling 21
Logging the Error 21
Only works for one error: 21
Rearms for multiple errors 21
Another example 22
Saving the Workbook in Different Forms 23
CSV 23
XLSX 23
XLS 23
XLSB 23
HTML 23
Which Workbook's Routines Get Called 23
Opening a Workbook (from another Workbook) 24
Opening a Password Protected Workbook 24
Getting a Sheet from the other Workbook 24
Database 25
Non Spreadsheet Sql 25
Selecting onto a Worksheet Sql 25
Misc Database Gotchas 25
Misc 26
Clipboard 26
Confirmation Prompts 26
Status Bar 26
Screen Updating (during macro run) 26
Line Break 26
Range 26
Array Starting Index 26
Max 26
Sql Server DateTime Format 26
NA 26
Alpha Cell Addressing to Numeric 27
CountA (Count No of Non Empty Cells in a Range) 27
CountIf 27
Typecasts 27
Other 27
Getting the Column Length 28
Getting the Max Row Length 28
Row, Column Numbers to Letters 28
Shell 28
Convert a column from text to numeric 29
Does File Exist 29
One line If Statement 29
Getting background to change depending on entry 29
Charts and Chart Coloring 30
To add a chart type of a different type 31
Second Axis 31

Excel Cheat Sheet

- Things I keep forgetting…

To select a whole column (up to the first blank cell)	Cntrl-Shift-<down arrow> Cntrl-Shift-<up arrow>
Counta(A1:A12)	Counts the number of non empty cells in a range
CountIf(C1:C12, ">150")	see CountIfs for multiple conditions

Null Comparison - Empty Cells Tricks

(This catches me out all the time)

```
Dim aOS, aCS as Variant

 aOS = Null
 aCS = ""

 If Len(aOS) = 0 Then       'false
   no_of_rows = 1
 End If

 If Len(aCS) = 0 Then       'true
   no_of_rows = 2
 End If

 If aOS = Null Then     'false!!!
   no_of_rows = 3
 End If

 If IsNull(aOS) Then    'true
   no_of_rows = 4
 End If
```

VLOOKUP

```
Dim sizeMode, aProd  as Variant
sizeMode = Application.WorksheetFunction.VLOOKUP(aProd, _
  Sheets(gREF_PROD_MODES_SHEET).Range("$A:$B"), 2, False)

Here's another variation that acts as an array formula

refProd                     weAreHere
-----------                 ---------
A   B     C                 A     B
Lot Moves Activities        Lot   =VLOOKUP($A:$A, refProd!$A:$C,2,False)

Copy col B of weAreHere down the column
the $A:$A acts an an array formula for whatWeWant
```

Formula VLOOKUP

```
For row = 2 To no_of_rows
  .Cells(row, 4).Formula = "=VLOOKUP($C" & row & "," & SHIP_OPERS_SHEET &
                                        "!$A:$F,6,FALSE)"
Next row
```

Programmatic VLOOKUP

```
For row = 2 To no_of_rows
  anOp = .Cells(row, 3)
  .Cells(row, 4) = Application.VLookup(anOp, _
                  Sheets(SHIP_OPERS_SHEET).Range("$A:$F"), 6, False)
Next row
```

Match

```
'aValue and match_result must be a variants for the match to work
match_result = Application.Match(aValue, _
  Sheets(gDOWNS_SHEET).Range("$E:$E"), 0)   '0=exact match, -1 less than, 1 gt

match_result has the row (starting at the range start = row 1), #N/A otherwise.
```

Note: I've had problems doing Match on date. Had to match the date contents of two cells. Just could not get a match on a variant holding the date. Work around is to use a typecast:

```
srceRow = Application.Match(CLng(theDate), _
                Sheets(gBUCKETS_SHEET).Range("$A:$A"), 0)

If IsNumeric(srceRow) Then  'is anything there for this date?
```

Note: I also had a problem with the second match in a subroutine. I had to use Val (CStr(aValue) did not work).

```
match_row = Application.Match(Val(wiplta_lot), _
      Sheets(gLOTS_DEV_SHEET).Range("$D:$D"), 0)
```

Correcting Match for the Real Row

```
Dim weekNo As Variant
    outRow = Application.Match(weekNo, .Range("A8:A20"), 0)
    If IsNumeric(outRow) Then
        outRow = outRow + 8 - 1  '=real row = Since we started at row A8
        .Cells(outRow, 4) = .Cells(inRow, BASECOL + 15)
```

Iterating through all the Worksheets

```
Dim aWorksheet As Worksheet
Dim aCollection As New Collection

  For Each aWorksheet In Worksheets
    If (aWorksheet.Name <> gERRORLOG_SHEET) And _
       (aWorksheet.Name <> gSUMMARY_SHEET) Then
      aCollection.Add Item:=aWorksheet.Name
    End If
  Next 'aWorksheet
```

SumIfs

```
=SUMIFS(
D2:D7,          what to sum
D2:D7,          what to check
">5")          the condition to check
```

In plain English it says: sum the values of the cells D2 to D7 if they are greater than 5. The result should be 27.

```
=SUMIFS(G2:G2190, K2:K2190, "=02-MAY-2011 *")
```

You don't need to include the equals sign, e.g. on the formula bar:

```
=SUMIFS($C:$C,$B:$B,"7")
```

is the same as

```
=SUMIFS($C:$C,$B:$B,"=7")
```

Sumifs Formula Bar Version

```
.Cells(no_of_rows + 1, 8).Formula = _
          "=SUMIFS(H2:H" & no_of_rows & ",B2:B" & no_of_rows & ", ""FREE"")"
```

Sumifs Code Version

```
.Cells(no_of_rows + 1, 8).Formula = _
   Application.WorksheetFunction.SumIfs(
             Range("H2:H" & no_of_rows),
             Range("B2:B" & no_of_rows),
             "FREE")
```

CountIf

```
  'Lot change.  how many of this lot are on the sheet?
  no_of_lots = WorksheetFunction.CountIf(Range("A:A"), thisLot)
```

AverageIfs

```
devPayG = Application.WorksheetFunction.AverageIfs(Range("D3:D" & no_of_rows), _
                                        Range("A3:A" & no_of_rows), _
                                        ">=" & startDateTime, _
                                        Range("A3:A" & no_of_rows), _
                                        "<" & endDateTime)
```

DateDiff

```
seconds_elapsed = DateDiff("s", startDate, Now)
```

Combining (And, Or) Logic

Wrong: This will run and give the **wrong** result:

```
=SUMIFS(
D2:D7,          what to sum
D2:D7,          what to check
">5 And <10")           the condition to check
```

Right:

```
=SUMIFS(
D2:D7,          what to sum
D2:D7,          what to check
">5",           the condition to check
D2:D7,          what to check
"<10")          the condition to check
```

Clearing Cells

```
'clear out the area from any former calculations
    .Range("BA1:BZ" & outRow).Unmerge
                            'do first,else might not be able to clear
    .Range("BA1:BZ" & outRow).ClearContents
    .Range("BA1:BZ" & outRow).NumberFormat = "General"
                            'else might have problems w/formula
```

another way:

```
    Sheets(worksheet_name).Select
    Cells.Select
    Selection.UnMerge 'do first, else might not be able to clear
    Selection.ClearContents
    Selection.Style = "Normal"
    Selection.NumberFormat = "General" 'else might have problems w/formula
```

Cell Style

Named Styles

.Cells(inRow, inCol).Style = "Bad"
`Selection.Style = "Normal"`

Bold

Set rgMatch = .Range(.Cells(1, 1), .Cells(1, 12))
rgMatch.Font.Bold = True

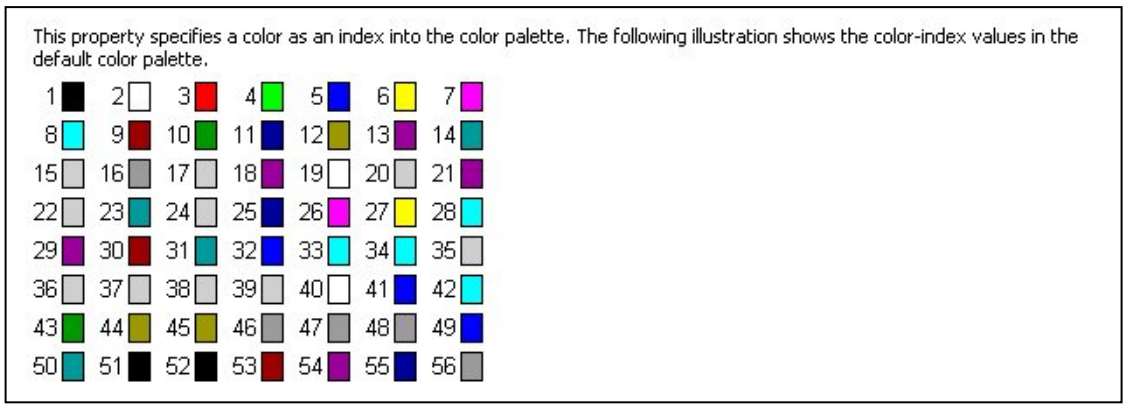
Font Foregrount

```
Set rgColor = .Range("A1", "Z75")
rgColor.Interior.ColorIndex = 53
```

Font Background

```
rgMatch.SpecialCells(xlCellTypeBlanks).Offset(, 1).Font.ColorIndex = 2
```

Default ColorIndex Pallet



Misc Styling

```
'Down tool lot?
If .Range("X" & row).Font.ColorIndex = 3 Then  'red font = down tool
  .Range("D" & row).Font.Strikethrough = True
End If

.Font.Underline = xlUnderlineStyleSingle
Font.Italic = True
```

Inserting Rows

Insert one blank row at a time at the destination. You can start with a filled out row and push it down to make a blank row

```
  Sheets(dest_worksheet_name).Rows("2:2").Select
  Application.CutCopyMode = False
  For i = 1 To no_of_srce_rows
    Selection.Insert shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
  Next i
```

To insert multiple blank rows all at once you must first clear out some blank rows first, since that selection of rows will be copied down to below itself

```
Sub merge_files(ByVal srce_workbook_name, _
                ByVal srce_worksheet_name, _
                ByVal dest_workbook_name, _
                ByVal dest_worksheet_name)

Dim no_of_srce_rows, no_of_dest_rows, i As Long
Dim j, insert_size As Long
```

```
    '-------------
    ' srce
    '-------------
    Workbooks(srce_workbook_name).Activate

    'delete the header row for easier selection
    Sheets(srce_worksheet_name).Rows("1:1").Select
    Selection.Delete Shift:=xlUp

    'Get how many rows in the srce we have to insert into the dest
workbook/worksheet
    no_of_srce_rows = get_column_length(srce_worksheet_name, "G", 2)

    'tweak the srce:
    'Convert column G (Qty=moves) from text to numeric (turns out this was
tricky)
    Range("G1:G" & no_of_srce_rows).Select
    With Selection
      Selection.NumberFormat = "General"
      .Value = .Value
    End With

    '-------------
    ' dest
    '-------------
    Workbooks(dest_workbook_name).Activate

    Sheets(dest_worksheet_name).Select
    Application.CutCopyMode = False

    'I think we are (memory?) limited on my little (1 gig memory) Asus EEEPC to
    'how many rows we can insert at once, so break up into chunks.

    insert_size = 100  'insert this many rows at a time, 200 is too many for
1gig memory
    If no_of_srce_rows > insert_size Then

      'Need to make room with some blanks
      Sheets(dest_worksheet_name).Rows("2:2").Select  'insert from row 2
downwards
      For i = 1 To insert_size
        Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
      Next i

      'Do the big chunks of insert_size rows
      j = (no_of_srce_rows \ insert_size) - 1  'vba: \ interger div, / floating
pt div
      Sheets(dest_worksheet_name).Rows("2:" & 1 + insert_size).Select  'row
1=header
      For i = 1 To j
        Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
      Next i
    End If

    'Do the modulus leftover to make room for the big chunks
    j = no_of_srce_rows Mod insert_size
    Sheets(dest_worksheet_name).Rows("2:2").Select  'insert from row 2
downwards
    For i = 1 To j
      Selection.Insert Shift:=xlDown, CopyOrigin:=xlFormatFromLeftOrAbove
    Next i
```

```
    'copy the srce to dest worksheet
    Workbooks(srce_workbook_name).Sheets(srce_worksheet_name).Rows("1:" &
no_of_srce_rows).Copy _

Destination:=Workbooks(dest_workbook_name).Sheets(dest_worksheet_name).Rows("2:
" & no_of_srce_rows)

    Application.CutCopyMode = False  'clear clipboard

End Sub  'merge_files
```

Deleting a Row

```
Sheets(gOPSTOOLS_SHEET).Rows("2:2").Delete Shift:=xlUp
```

Delete a row if an entire column is blank

```
Sheets(gOPSTOOLS_SHEET).Range("$C:$C").Select
Selection.SpecialCells(xlCellTypeBlanks).EntireRow.Delete
```

Another way

```
Dim rgOutput As Range
    iLastRow = .Cells(.Rows.Count, 1).End(xlUp).row
    Set rgOutput = .Range("A1").Resize(iLastRow, iLotsAllCols)
    On Error Resume Next  'In case there are no blank cells
    rgOutput.SpecialCells(xlCellTypeBlanks).EntireRow.Delete Shift:=xlUp
    On Error GoTo 0
```

Delete a row if the entire row is blank

```
dim row as long
    With Sheets(sheetName)
      'Delete blank rows
      For row = MAX_ROWS To 1 Step -1
        If WorksheetFunction.CountA(.Rows(row)) = 0 Then
            .Rows(row).EntireRow.Delete
        End If
      Next row
    End With
```

Copying a row between Worksheets

```
Sheets(gOPSTOOLS_SHEET).Rows(match_row & ":" & match_row).Copy _
            Destination:=Sheets(gOPSDOWN_SHEET).Rows(row & ":" & row)
```

Determining Worksheet Size

Rows

```
Function get_column_length(ByVal worksheet_name As String, _
                           ByVal column As Variant, _
                           Optional ByVal startRow As Integer = 1)
Dim inRow, inCol As Long
    inRow = startRow
    With Sheets(worksheet_name)
      If IsNumeric(column) Then 'use cells
        inCol = column  'need a Long (not Variant) to use with .Cells
        Do While (Len(.Cells(inRow, inCol).Value) > 0)
          inRow = inRow + 1
        Loop
      Else                        'use range
        Do While (Len(.Range(column & inRow).Value) > 0)
          inRow = inRow + 1
        Loop
        'Note: tried this one liner alternative, but sometimes picks up too much
        'get_column_length = WorksheetFunction.CountA(Range(column_letter &
"1").EntireColumn)
      End If
    End With

    inRow = inRow - 1
    get_column_length = inRow

End Function  'get_column_length
```

Columns

```
Function get_row_length(ByVal worksheet_name As String, ByVal row As Long)
Dim LastCell As Range, RowLength As Long

   With Sheets(worksheet_name)
     With Cells(row, 1).EntireRow
       Set LastCell = .Cells(row, .Columns.Count).End(xlToLeft)
     End With
   End With

   RowLength = 1 + LastCell.column
   get_row_length = RowLength

End Function
```

Another way

```
iNextCol = .Cells(1, .Cells.Columns.Count).End(xlToLeft).column + 2
```

Copying just the Visible (filtered) Cells

```
     'Filter the results sheet, so it displays only this 828 owner
     Sheets(gRESULTS_SHEET).Select
     Cells.Select
     Selection.AutoFilter
     ActiveSheet.Range("$A$1:$CZ$" & no_of_results_rows).AutoFilter Field:=3, _
       Criteria1:=owner828

     'Select and copy these filtered cells over to a temp sheet
     'for easier date searching for just these owner's rows from
```

```
    'the results sheet

    Selection.SpecialCells(xlCellTypeVisible).Select  'select only visible rows
    Selection.Copy

    clear_a_worksheet (TEMP_SHEET)
    Sheets(TEMP_SHEET).Range("A1").Select
    ActiveSheet.Paste
```

Copying between Workbooks

```
    wb.Sheets(srce_worksheet).Range("A1:CZ" & no_of_rows).Copy _
      Destination:=ThisWorkbook.Sheets(dest_worksheet).Range("A1:CZ" & no_of_rows)
```

Deleting worksheets

Deleting All Worksheets of a Type

```
Dim aWorksheet As Worksheet

 'Clear out the old owner worksheets before making new ones
 Workbooks(ThisWorkbook.Name).Activate

For Each aWorksheet In Worksheets

    If (Left(aWorksheet.Name, 3) <> "sum") And _
       (Left(aWorksheet.Name, 3) <> "ref") Then

      delete_a_worksheet (aWorksheet.Name)

    End If
  Next 'aWorksheet
```

Deleting Worksheets that Might Not Exist

```
Function does_worksheet_exist(ByVal wksName As String) As Boolean
   On Error Resume Next
   does_worksheet_exist = CBool(Len(Worksheets(wksName).Name) > 0)
   On Error GoTo 0
End Function
'
' delete the worksheet name passed in
' (if it exists)
'
Sub delete_a_worksheet(ByVal worksheet_name As String)

  If (does_worksheet_exist(worksheet_name)) Then
    Sheets(worksheet_name).Select
    Application.DisplayAlerts = False  'no confirmation prompts
    ActiveWindow.SelectedSheets.Delete
    Application.DisplayAlerts = True
  End If

End Sub
```

Does Worksheet Exist

Method 1

```
Dim wsNew As Worksheet

   On Error Resume Next
   'Check to see if Sheets called "Flow" & "RouteFlow" exist.
   'If not, create them.
   Set wsNew = wBook.Worksheets("Flow")
   If wsNew Is Nothing Then
      Set wsFlow = wBook.Worksheets.Add
      wsFlow.Name = "Flow"
   End If
```

Method 2

```
'*************************************************************
' Does the worksheet name passed in exist?
' Return True if so, False otherwise
'
Function does_worksheet_exist(ByVal wksName As String) As Boolean
   On Error Resume Next
   does_worksheet_exist = CBool(Len(Worksheets(wksName).Name) > 0)
   On Error GoTo 0
End Function
```

Date Time

```
Dim aDate, aDateTime As Variant
Dim strMonth, strDay, wildCardName As String

aDateTime = Now   ‘returns   mm/dd/yy hh:mm:ss

aDate = Date   ‘returns current date  m/d/yyyy

strYear = DatePart("yyyy”, aDate)  ‘ y is day of year

strYear = Trim(Str(Int(DatePart("yyyy", aDate)) - 2000))   ‘2 digit year

 aDate = DateAdd("d", -1, Date)  'go back -1 many days

 strMonth = DatePart("m", aDate)
 strDay = DatePart("d", aDate)
 strDay = Format(strDay, "0#") 'add leading zero

 wildCardName = strMonth & strDay & "_*"

 aDate = DateAdd("d", -gWIPLOT_DAYS_AGO, Date)  'go back these many days
 strYear = DatePart("yyyy", aDate)
 strMonth = DatePart("m", aDate)
 strDay = DatePart("d", aDate)

 longYear = Int(strYear)
 longMonth = Int(strMonth)
 longDay = Int(strDay)
 startDate = (longYear * 10000) + (longMonth * 100) + longDay

DateDiff("d", 0, .Range("A" & i)))
DateDiff("d", oldDate, newDate)

'extract out the day-month (eg 3-May) from the timestamp
strMonth = MonthName(DatePart("m", .Cells(outRow, 9).Value))
strDay = DatePart("d", .Cells(outRow, 9).Value)

another way to do:
… = MonthName(Month(.Cells(runningRow, 1)))


 'Change the date column to a date format of (eg) 31-Mar-11
.Range("A:A").NumberFormat = "[$-409]d-mmm-yy;@"


’06-May-2011
        strMonth = Left(MonthName(DatePart("m", Now)), 3)
        strDay = DatePart("d", yesterdayDate)
        strDay = Format(strDay, "0#") 'add leading zero
        strYear = DatePart("yyyy", yesterdayDate)  ' y is day of year

        'cost report date eg: 06-MAY-2011
        costReportDate = strDay & "-" & strMonth & "-" & strYear
```

Common Block of Date Code

```
Dim yyyymmdd_yesterday As String
Dim aYear, aDay, aMonth As String
Dim aDateToday, aDateYesterday As Date

  'yyyymmdd gets TODAYs sheet which has YESTERDAYs data
  'Subtract one day from this
  aYear = Left(yyyymmdd_today, 4)
  aMonth = Mid(yyyymmdd_today, 5, 2)
  aDay = Right(yyyymmdd_today, 2)

  aDateToday = aMonth & "/" & aDay & "/" & aYear

  aDateYesterday = DateAdd("d", -1, aDateToday)

  aYear = DatePart("yyyy", aDateYesterday)
  aMonth = Format(DatePart("m", aDateYesterday), "0#")   'add leading zero too
  aDay = Format(DatePart("d", aDateYesterday), "0#") 'add leading zero too

  yyyymmdd_yesterday = aYear & aMonth & aDay
```

Time

```
Dim aDate as Variant
Dim strHour, strMin, strSec as String

      aDate = Now
      strHour = DatePart("h", aDate)
      strMin = DatePart("n", aDate)
      strSec = DatePart("s", aDate)

      strHour = Format(strHour, "0#") 'add leading zero
      strMin = Format(strMin, "0#")   'add leading zero
      strSec = Format(strSec, "0#")   'add leading zero
```

Convert datetime column to just display the date

'convert the datetime column to just display the date

```
  Sheets("running").Columns("A:A").Select
  Range("A2").Activate
  Selection.NumberFormat = "mm/dd/yy;@"
```

Weekday

Dim enumDay As Integer
enumDay = Weekday(Now, vbSunday) '1=Sumday 7=Saturday

Range

Non contiguous
```
Range("A1:A10,C1:C10,E1:E10")
```

Square brackets style
```
[A1:A10,C1:C10,E1:E10]
```

Referring to a non active worksheet
```
Worksheets("Sheet1").Range("C10")
```

Referring to a non active workbook
```
Workbooks("Sales.xls").Worksheets("Sheet1").Range("C10")
```

With cells
```
Range(Cells(1,1), Cells(10,5))
```

Using a Range object
```
Dim srce_rng, dest_rng As Range
Set srce_rng = Sheets(gREADIN_DATA_SHEET).Range("$A$1:$Z$" & no_of_rows)
```

Remember - the Range object isn't just the range. It includes the workbook and sheet
```
srce_rng.Select
```

Column Number to Letter

```
'*************************************************************
' e.g. column 5 returns "E"
'*************************************************************
Function column_number_to_alpha(colNumber As Long) As String

Dim alphaCell As String
Dim pieces As Variant

  alphaCell = Cells(1, colNumber).Address  'returns (eg) $DK$41
  pieces = Split(alphaLetter, "$")
  column_number_to_alpha = pieces(1)

End Function 'column_number_to_alpha
```

Subroutines & Functions

Function calls always get parenthesis,

```
my_number = aFunction(someParm)
```

Functions cannot manipulate worksheets

Parenthesis can only be used with one parm for subroutines

```
aSub (parm1)
aSub parm1

aSub parm1, parm2
```

There's an optional Call keyword for calling subroutines:

```
Call aSub(parm1, parm2)
```

Optional Subroutine Parm

```
Function get_column_length(ByVal worksheet_name As String, _
                           ByRef column As Variant, _
                  Optional ByVal startRow As Long = 1)
```

System Calls & Time Delay

System calls are asynchronous.
Making them synchronous draws in a lot of operating system code
It's simplier to just wait (with a time delay) instead:

```
newHour = Hour(Now())
newMinute = Minute(Now())
newSecond = Second(Now()) + 10
waitTime = TimeSerial(newHour, newMinute, newSecond)
Application.Wait waitTime

So if you want to delay 1 minute then
Application.Wait (now() + timevalue("00:01:00"))
```

Alternative way to do:

```
    Application.Wait Now() + TimeSerial(0, 0, 0.9)
```

Arrays

```
Cannot do a constant array in VBA

Sub test_move_tgts()
  Dim targets_array(1 To 13) As Long
  get_owner_move_targets "Carmen", "PAYG", targets_array
End Sub

Sub get_owner_move_targets(ByVal anOwnerName As Variant, _
                           ByVal anOwnerCat As Variant, _
                           ByRef targets_array() As Long)
  …
End Sub

Dim srceCell, destCell As Variant
    srceCell = Array("I2", "J2", "K2", "L2", "M2", "N2")
    destCell = Array("D11", "D4", "D9", "D13", "D3", "D7")

Size of Array

  Dim aSize as long
  aSize = UBound(srceCell) + 1  'UBound is the highest index!


'Array indexes start at 0 unless Base 1 specified
'UBound does NOT give the array size.

    For i = 0 To UBound(srceCell)
      Range(srceCell(i)).Select
…
```

Split

Splitting off filename from path

```
    Dim pieces As Variant
    pieces = Split(srce_file_name, "\")
```

Getting last array element

```
    srce_workbook_name = pieces(UBound(pieces))
```

Collections

```
Dim aCollection as New Collection
Dim aValue as Variant  'must be variant
'since collections often arrange elements alphabetically you
'must delimit pairs (rather than use two corresponding collections as you might
'with arrays)
aCollection.Add Item:= aName & ";" & aNumber  'note the ; delimiter

For Each aValue in aCollection
  pieces = split(aValue, ";")    'use split to undelimit
  aName = pieces(0)
  aNumber = pices(1)
…
Next  'aValue

set aCollection = Nothing 'clear out collection for reuse
```

Turning a Column into a Collection

```
Dim rng as Range
Dim aValue as Variant

'The sheet must be selected before assigning the range
Sheets(gCALCDOWNS_SHEET).Select

Set rng = Range("$B2:$B500")

For Each aValue in rng
  ...
Next
```

Error Handling

```
Public Function file_exists(ByVal strFullPath As String) As Boolean
    On Error GoTo EarlyExit
    If Not Dir(strFullPath, vbDirectory) = vbNullString Then file_exists = True

EarlyExit:
    On Error GoTo 0
End Function
```

Careful with loops: once you hit the error goto - it might not be active for the next iteration of the loop

Logging the Error

EarlyExit:
msgbox "The error was " & Err.Description

Only works for one error:

```
       On Error GoTo allZeros
       For outCol = 14 To 17
         .Cells(outRow, outCol) = _
            Application.WorksheetFunction.AverageIfs(Range(.Cells(3, outCol),
                  .Cells(no_of_owner_rows, outCol)), Range(.Cells(3, outCol),
                  .Cells(no_of_owner_rows, outCol)), ">0")

         'Tack the average onto the column legend
         If IsNumeric(.Cells(outRow, outCol)) Then 'were there any non zero
numbers?
           .Cells(1, outCol) = .Cells(1, outCol) & "=" & .Cells(outRow, outCol)
         Else
           .Cells(1, outCol) = .Cells(1, outCol) & "=0"
         End If

         GoTo nextCol

allZeros:  .Cells(1, outCol) = .Cells(1, outCol) & "=0"
nextCol:
```

Rearms for multiple errors

```
       For outCol = 14 To 17
         On Error GoTo allZeros
         .Cells(outRow, outCol) = _
            Application.WorksheetFunction.AverageIfs(Range(.Cells(3, outCol),
                  .Cells(no_of_owner_rows, outCol)), Range(.Cells(3, outCol),
                  .Cells(no_of_owner_rows, outCol)), ">0")
         'Tack the average onto the column legend
         If IsNumeric(.Cells(outRow, outCol)) Then 'were there any non zero
numbers?
           .Cells(1, outCol) = .Cells(1, outCol) & "=" & .Cells(outRow, outCol)
         Else
           .Cells(1, outCol) = .Cells(1, outCol) & "=0"
         End If

         GoTo nextCol

allZeros:  .Cells(1, outCol) = .Cells(1, outCol) & "=0"
```

```
nextCol: On Error GoTo 0  'rearms for multiple errors

       Next outCol
```

Another example

```
      Application.DisplayAlerts = False  'no confirmation prompts
      On Error GoTo skipit1 'else we might delete whatever other sheetname is
selected
      Sheets("Chart_WIP").Visible = True  'cannot delete it otherwise
      Sheets("Chart_WIP").Delete
      GoTo skipit2    'can only use a resume in an error handler

skipit1:
      Resume skipit2  'must use a resume to exit the first error handler

skipit2:
      On Error GoTo skipit3
      Sheets("Chart_Wafers").Visible = True  'cannot delete it otherwise
      Sheets("Chart_Wafers").Delete
      GoTo skipit4

skipit3:
      Resume skipit4

skipit4:
      On Error GoTo 0
      Application.DisplayAlerts = True  'turn confirmation prompts back on
```

Saving the Workbook in Different Forms

```
ActiveWorkbook.Save      'Make sure the original is saved with the latest updates
```

CSV

```
  Dim pieces as Variant
  Dim destFile as String

  pieces = Split(ThisWorkbook.Name, ".")
  destFile = gHOME_DIR & pieces(0) & ".csv"

  Application.DisplayAlerts = False  'no prompts, allow overlaying old stuff
  ActiveWorkbook.SaveAs Filename:=destFile,FileFormat:=xlCSV,
_CreateBackup:=False
```

XLSX

Workbook Without Macros

```
  destFile = "c:\inetpub\wwwroot\svtc\mov_summary.xlsx"
  Application.DisplayAlerts = False
  ActiveWorkbook.SaveAs filename:=destFile, FileFormat:=xlWorkbookDefault
```

XLS

Workbook Without Macros

```
  destFile = "c:\inetpub\wwwroot\svtc\Daily_Cost_Report.xls"
  Application.DisplayAlerts = False
  ActiveWorkbook.SaveAs filename:=destFile, FileFormat:=xlWorkbookNormal
```

XLSB

```
  destFile = "c:\inetpub\wwwroot\svtc\Daily_Cost_Report.xlsb"
  Application.DisplayAlerts = False
  ActiveWorkbook.SaveAs filename:=destFile, FileFormat:=xlExcel12
```

HTML

```
  destFile = "c:\inetpub\wwwroot\svtc\Daily_Cost_Report.htm"
  Application.DisplayAlerts = False
  ActiveWorkbook.SaveAs filename:=destFile, FileFormat:=xlHtml
```

Which Workbook's Routines Get Called

```
 '-----------------------------------------------
  'Vector away into the workbook we just opened
  '-----------------------------------------------
  Workbooks(wb.Name).Activate

  '-------------------------------------------------------
  'First button push:
  ' Fab2 sheet button: "Finish Data Entry, Update Summary"
  '-------------------------------------------------------
  Sheets(gFAB2_SHEET).Activate

  Application.Run ThisWorkbook.Name & "!lockAllells"  'must be on gFAB2_SHEET before calling this
```

Opening a Workbook (from another Workbook)

Good

Set wb = Workbooks.Open(gWEB_DIR & srce_workbook, False, True) 'updatelinks, read only

Better

Workbooks.Open Filename:=gWEB_DIR & srce_workbook, UpdateLinks:=False, ReadOnly:=True

Opening a Password Protected Workbook

```
'--- old non password protected workbook code ---
'                                       UpdateLinks, ReadOnly
Set wb = Workbooks.Open(local_master_file, False, False)
Workbooks(wb.Name).Activate

'--- new password protected workbook code ---
Workbooks.Open Filename:=local_master_file, UpdateLinks:=0, _
                      Password:=gPW, WriteResPassword:=gPW
Workbooks(master_file_name).Activate
Set wb = ActiveWorkbook
```

Getting a Sheet from the other Workbook

```
'Copy the srce_worksheet to the dest_worksheet in ThisWorkbook
Dim no_of_rows as Long
        Workbooks(ThisWorkbook.Name).Activate

        Workbooks(srce_workbook).Sheets(aSheetName). _
            Range("A1:CZ" & no_of_rows).Copy _
            Destination:=ThisWorkbook.Sheets(aSheetName). _
            Range("A1:CZ" & no_of_rows)
```

Database

Non Spreadsheet Sql

```
'*************************************************************
' Note: To use this you must include
'       Microsoft Active X 2.8 Data Objects Library
'       via Tools->References from the VB Editor
'*************************************************************
Sub run_sql(stSql)
   Dim thisCon As ADODB.Connection
   Set thisCon = New ADODB.Connection
   thisCon.Open gODBC_NAME, gUSER, gPW
   thisCon.Execute (stSql)
   thisCon.Close
End Sub    'run_sql
```

Selecting onto a Worksheet Sql

```
Sub stores_move()
Dim stSql As String
    If (Len(gStConn) <= 0) Then 'incase we run this as stand alone
      init_gStConn
    End If

    stSql = ""
    stSql = stSql & "SELECT DISTINCT WIPLOT.OWNER "
    stSql = stSql & "FROM WIPLOT WIPLOT "
    stSql = stSql & "WHERE WIPLOT.FACILITY='FAB1A' "
    stSql = stSql & "AND WIPLOT.WORKSTREAM_ERASE_FLAG Is Null "

    On Error GoTo sql_err
    With Sheets(OWNERS_SHEET)
      With .QueryTables.Add(Connection:=gStConn, Destination:=.Cells(1, 1))
          .CommandText = stSql
          .FieldNames = True
          .BackgroundQuery = False
          .RefreshStyle = xlOverwriteCells
          .Refresh BackgroundQuery:=False
          .Delete
      End With
    End With
    On Error GoTo 0
    GoTo no_err

sql_err:
    log_error_msg "ERROR add_lots_dev_os_cs_gatingLot query: " &
Err.Description
    On Error GoTo 0
no_err:

End Sub
```

Misc Database Gotchas

Too many items in an IN ('ccc','ddd'…). Have to break down to multiple INs

Misc

Clipboard

```
Application.CutCopyMode = False 'clear clipboard
```

Confirmation Prompts

```
Application.DisplayAlerts = False  'no confirmation prompts (for saving, etc.)
```

Status Bar

```
Application.StatusBar = "Getting cost report moves for " & todayDate
Application.StatusBar =  False
```

Screen Updating (during macro run)

```
Application.ScreenUpdating = False
```

Line Break

vbCr
vbLf
vbNewLine

```
ActiveChart.ChartTitle.Text = owner_name & vbNewLine & "Moves and Activities"
```

Range

```
Dim sumRange As Range

Set sumRange = .Range(.Cells(3, 10 + i), .Cells(no_of_rows, 10 + i))
```

Array Starting Index

Option Explicit
Option Base 1
The element's index of the array starts from 0 unless **Option Base 1** is specified in the public area (area outside of the sub procedure). If Option Base 1 is specified, the index will start from 1.

Max

```
  max_run_payg = Application.WorksheetFunction.Max(Range("BV:BV"))

  ActiveChart.Axes(xlValue).MinimumScale = 0
  ActiveChart.Axes(xlValue).MaximumScale = yaxis_max
```

Sql Server DateTime Format

```
WHERE (hours.date>{ts '2010-12-14 08:21:38'})
```

NA

=na() returns #N/A

To check a cell for #N/A

```
If   IsError(.Cells(i, 12).Value) Then
```

Alpha Cell Addressing to Numeric

ROW(C10) 'returns 10

COLUMN(D10) 'returns 4

CountA (Count No of Non Empty Cells in a Range)

Counta(A1:A12)

CountIf

CountIf(C1:C12, ">150") 'see countifs for multiple conditions

Typecasts

CBool(expression)
CByte(expression)
CCur(expression)
CDate(expression)
CDbl(expression)
CDec(expression)
CInt(expression)
CLng(expression)
CSng(expression)
CVar(expression)
CStr(expression)

Other

```
Workbooks(ThisWorkbook.Name).Activate
```

- ThisWorkbook refers to the workbook that's running the current code.

You cant do a select (eg Range select) inside a "with"
eg:

```
With sheets(aSheet)
  .Range("A1:C3").Select
```

…cant do this

Getting the Column Length

```
 'Note that there might be blank rows in this spreadsheet
 no_of_rows = Sheets(sheet_name).Range("A3000").End(xlUp).row  '3000 lots max
```

Getting the Max Row Length

```
no_of_columns  =  Sheets(sheet_name).Range("A3000").End(xlToRight).row
```

Row, Column Numbers to Letters

```
Dim max_cell As String

  max_cell = Sheets("Lots_CFP").Cells(no_of_rows, no_of_columns).Address
```

'returns (eg) $DK$41

Shell

aka: system, exec

```
    retc = Shell("c:\perl\bin\perl.exe
c:\svtc\kencode\perl_scripts\get_cost_report.pl", _
    vbNormalFocus)  'returns the task id, so we cant use it for much

    'Since the above Shell command runs asynchronously, delay a few seconds.
    '(Making it synchronous is a big deal)

    do_seconds_delay (20) 'should be enough '7/20/2011 5 seconds to 20 seconds

    'is todays xml file there?
    If file_exists(full_filename) Then
      cost_report_get_file = full_filename
    Else
      cost_report_get_file = ""
    End If
```

Convert a column from text to numeric

```
'Convert column G from text to numeric (turns out this was tricky)
Range("G2:G" & no_of_rows).Select
With Selection
  Selection.NumberFormat = "General"
  .Value = .Value
End With
```

Does File Exist

One line If Statement

```
Public Function does_file_exist(ByVal strFullPath As String) As Boolean

 On Error GoTo EarlyExit
 If Not Dir(strFullPath, vbDirectory) = vbNullString Then dodes_file_exist =
True

EarlyExit:
    On Error GoTo 0

End Function

If MsgBox("Run the macro?", vbYesNo) = vbNo Then Exit Sub
```

Using object vs object name

Workbooks(wb.name).
vs
wb.

Getting background to change depending on entry

Without using macro code

Review -> Unprotect Sheet
select a new cell
Home -> Consitional Formatting
  White (background and font) for errors
    - Init the cells with "=NA()", which will be an error
  Use other conditional formatting colors

You activate workbooks but Select sheets

Workbooks(ThisWorkbook.Name).Activate

Charts and Chart Coloring

Chart Types:
xlColumnStacked,
xlColumnClustered,
xlLineMarkers

```
    ActiveSheet.Shapes.AddChart.Select
    ActiveChart.chartType = xlColumnStacked
    ActiveChart.SeriesCollection.NewSeries
    ActiveChart.SeriesCollection(1).Name = Range("$AE$1")
    ActiveChart.SeriesCollection(1).Values = Range("$AE$2:$AE$" & outRow)
    ActiveChart.SeriesCollection(1).Interior.Color = RGB(0, 255, 0)  'rgb

    'make goal line a little thicker
    ActiveChart.SeriesCollection(5).Border.Weight = xlThick
                  'xlHairline, xlThin, xlMedium & xlThick

    'Pretty up the chart
    ActiveChart.Axes(xlCategory).TickLabels.Orientation = 90 'rotate the text
this many degrees


Dim RngToCover As Range
Dim ChtOb As ChartObject

    'Position the chart
    Set RngToCover = ActiveSheet.Range("B90:H105")
    Set ChtOb = ActiveChart.Parent
    ChtOb.Height = RngToCover.Height ' resize
    ChtOb.Width = RngToCover.Width   ' resize
    ChtOb.Top = RngToCover.Top       ' reposition
    ChtOb.Left = RngToCover.Left     ' reposition

    If chartType = xlColumnClustered Then  'barchart

      ActiveChart.SeriesCollection.NewSeries
      ActiveChart.SeriesCollection(1).Values = Range("$AR$2:$AR$" & outRow)
      ActiveChart.SeriesCollection(1).Interior.Color = RGB(74, 130, 189)

      ActiveChart.SeriesCollection.NewSeries
      ActiveChart.SeriesCollection(2).Name = Range("$AS$1")
      ActiveChart.SeriesCollection(2).Values = Range("$AS$2")

      ActiveChart.SeriesCollection(2).XValues = Range("$AI$2:$AI$" & outRow)
      ActiveChart.SeriesCollection(2).Interior.Color = RGB(74, 130, 189)

    If chartType = xlLineMarkers Then  'line chart

    ActiveChart.SeriesCollection(1).Border.Color = RGB(255, 151, 67)
    ActiveChart.SeriesCollection(1).MarkerForegroundColor = RGB(255, 151, 67)
    ActiveChart.SeriesCollection(1).MarkerBackgroundColor = RGB(255, 151, 67)
```

To add a chart type of a different type

```
    ActiveChart.SeriesCollection.NewSeries
    ActiveChart.SeriesCollection(2).chartType = xlLineMarkers
```

Second Axis

```
    ActiveChart.SeriesCollection(4).AxisGroup = 2  'axis on right

    'XAxis day
    ActiveChart.SeriesCollection(1).XValues = Range("$A$3:$A$" & outRow)
    ActiveChart.Axes(xlCategory).TickLabels.Orientation = -90
                          'rotate the text this many degrees
    ActiveChart.ApplyLayout (1)

    ActiveChart.ChartTitle.Select
    ActiveChart.ChartTitle.Text = owner_name & " Actual M/I" & vbCr & "(Cost
Report) Moves per (Daily Average) Inventory"
    ActiveChart.ChartTitle.Font.Size = 14
  ActiveChart.Axes(xlValue).HasTitle = True
  ActiveChart.Axes(xlValue).AxisTitle.Text = "Inventory (Wafers)"

  ActiveChart.Axes(xlValue, xlSecondary).HasTitle = True
  ActiveChart.Axes(xlValue, xlSecondary).AxisTitle.Text = "Moves/Inventory"
```