My Company

Tel: +44 1234 567 9898
Fax: +44 1234 545 9999
email: info@@company.com

Microsoft Excel VBA

Source Code Output

Created using

VBAcodePrint

Macro Variables Substitution Example

General Date	14/01/2017 9:23:52 PM
Long Date	Saturday, 14 January 2017
Short Date	14/01/2017
Long Time	9:23:52 PM
Short Time	21:23
Today	14/01/2017 9:23:52 PM
Project Name	VBAProject

Prepared by
Joginder S Nahil
on
14/01/2017 9:23:52 PM

WWW.STARPRINTTOOLS.COM

```
1   Option Explicit
2   ' Module variables
3   Public Const mcdblVersion As Double = 20
4   Public Const mcstrMacrosPath As String = "\bgss09\is\service management release
    management\excel macros"
5
6   Dim mintNextRow As Integer
7   Dim mzGroupRows As Collection
8   Dim mzRotaSheet As Worksheet
9
10  '================================================================================
11  '================================================================================
12  '================================================================================
13  '================================================================================
14  '================================================================================
15  ' DEPENDENCY BUILDING
16  '================================================================================
17  '================================================================================
18  '================================================================================
19  '================================================================================
20  '================================================================================
21
22  Sub AddDependency()
23      ' Called when we want to add a dependency
24      ' Assign to Ctrl+Q
25      BuildDependency (False)
26  End Sub
```

```
27
28      Sub BuildDependency(ByVal rblnReplace As Boolean)
29          ' Only interested in implementation sheet
30          If UCase(ActiveSheet.Name) <> "IMPLEMENTATION"  Then
31              MsgBox "Can only build dependencies on the Implementation sheet"
32              Exit Sub
33          End If
34
35          ' Go through the selected cells
36          ' Any with a column of 1 are those that we wish to make dependent
37          ' Any with a column of 7 are those which need to have the dependency
38
39          ' First we need to get a list of the dependencies
40          Dim zCell As Range
41          Dim intDependentCells(20) As Integer
42          Dim intDependentCellIx As Integer
43          intDependentCellIx = 0
44          For Each zCell In Selection.Cells
45              If zCell.Column = 1 Then
46                  intDependentCells(intDependentCellIx) = zCell.row
47                  intDependentCellIx = intDependentCellIx + 1
48              End If
49          Next
50
51          ' REMOVED FOR V3
52          ' No dependencies, so what's the point
53          ' If intDependentCellIx = 0 Then Exit Sub
54
55          ' Now we need to go through and write the dependencies
56          For Each zCell In Selection.Cells
57              If zCell.Column = 7 Then
58                  Dim ix As Integer
59                  Dim strDepList As String
60                  strDepList = ""
61                  Dim strStart As String
62                  strStart = ""
63                  If intDependentCellIx = 0 Then
64                      ' No dependencies selected so we default to the previous task
65                      strDepList = strDepList + "TEXT(A"  & zCell.row - 1 & "," "0.000" ")"
66                      strStart = strStart & "F"  & zCell.row - 1
67                  Else
68                      For ix = 0 To intDependentCellIx - 1
69                          If ix > 0 Then
70                              strDepList = strDepList & "&" ", " "&"
71                              strStart = strStart & ","
72                          End If
73                          strDepList = strDepList + "TEXT(A"  & intDependentCells(ix) & "," "0.000"
                            ")"
74                          strStart = strStart & "F"  & intDependentCells(ix)
75                      Next
76                  End If
77                  If rblnReplace Or zCell.Formula = ""  Then
78                      zCell.Formula = "="  & strDepList
79                      ActiveSheet.Range("E"  & zCell.row).Formula = "=MAX("  & strStart & ")"
80                  Else
81                      ' Existing single reference?
    1 2 3 4
```

```
1 2 3 4
82              If Left(zCell.Formula, 2) = "=A" Then
83                  ' Existing formula?
84                  zCell.Formula = "=TEXT(" & Right(zCell.Formula, Len(zCell.Formula) - 1) &
                    "," "0.000" ")&" ", " "&" & strDepList
85              ElseIf Left(zCell.Formula, 1) = "=" Then
86                  zCell.Formula = zCell.Formula & "&" ", " "&" & strDepList
87              Else ' Must just be text so encapsulate
88                  zCell.Formula = "=" "" & zCell.Text & ", " "&" & strDepList
89              End If
90              Dim strExistingStartFormula As String
91              strExistingStartFormula = UCase(Trim(ActiveSheet.Range("E" & zCell.row).Formula
                ))
92              Dim strNewStartFormula As String
93              If Left(strExistingStartFormula, 5) = "=MAX(" Then
94                  strExistingStartFormula = Mid(strExistingStartFormula, 6, Len(
                    strExistingStartFormula) - 6)
95              ElseIf Left(strExistingStartFormula, 1) = "=" Then
96                  strExistingStartFormula = Mid(strExistingStartFormula, 2, Len(
                    strExistingStartFormula) - 1)
97              Else ' nothing to add to
98                  strExistingStartFormula = "" "" & Trim(ActiveSheet.Range("E" & zCell.row).
                    Value) & "" ""
99              End If
100             strNewStartFormula = "=MAX(" & strExistingStartFormula & "," & strStart & ")"

101             ActiveSheet.Range("E" & zCell.row).Formula = strNewStartFormula
102         End If
103     End If
104 Next
105 End Sub
```

```
106
107     '=============================================================================
108     '=============================================================================
109     '=============================================================================
110     '=============================================================================
111     '=============================================================================
112     ' ROTA CONSTRUCTION
113     '=============================================================================
114     '=============================================================================
115     '=============================================================================
116     '=============================================================================
117     '=============================================================================
118
119     Public Sub ConstructRota()
120
121       ' Need to make sure placeholders are displayed otherwise we have problems
122       ' resizing comments (macro gets a stupid error)
123       If ActiveWorkbook.DisplayDrawingObjects = xlHide Then
124         ActiveWorkbook.DisplayDrawingObjects = xlPlaceholders
125       End If
126
127       ' Validate the status quo
128       If Not SheetExists("Implementation" ) Then
129         MsgBox "The main plan must be in a sheet called 'Implementation'"  & vbCrLf & vbCrLf
          & "Unable to continue" , vbCritical, "Error"
130         Exit Sub
131       End If
132
133       Dim zImplSheet As Worksheet
134       Set zImplSheet = Sheets("Implementation" )
135
136       If Not (Trim(UCase(zImplSheet.Range("E1" ).Value)) = "PLANNED START" ) Then
137         zImplSheet.Range("E1" ).Select
138         MsgBox "Column E must contain the planned start date & time and be titled 'Planned
          Start'"  & vbCrLf & vbCrLf & "Unable to continue" , vbCritical, "Error"
139         Exit Sub
140       End If
141
142       If Not (Trim(UCase(zImplSheet.Range("F1" ).Value)) = "PLANNED END" ) Then
143         zImplSheet.Range("F1" ).Select
144         MsgBox "Column F must contain the planned end date & time and be titled 'Planned
          End'"  & vbCrLf & vbCrLf & "Unable to continue" , vbCritical, "Error"
145         Exit Sub
146       End If
147
148       If Not (Trim(UCase(zImplSheet.Range("H1" ).Value)) = "RESPONSIBILITY" ) Then
149         zImplSheet.Range("H1" ).Select
150         MsgBox "Column H must contain the planned end date & time and be titled
          'Responsibility'"  & vbCrLf & vbCrLf & "Unable to continue" , vbCritical, "Error"
151         Exit Sub
152       End If
153
154       If Not (Trim(UCase(zImplSheet.Range("I1" ).Value)) = "SUPPORT" ) Then
155         zImplSheet.Range("I1" ).Select
156         MsgBox "Column I must contain the planned end date & time and be titled 'Support'"  &
           vbCrLf & vbCrLf & "Unable to continue" , vbCritical, "Error"
1 2
```

```
1 2
157         Exit Sub
158     End If
159
160     ' Don't want any nasty alerts
161     Application.DisplayAlerts = False
162
163     Dim strStart As String
164     Dim strEnd As String
165     Dim strSlotLength As String
166     ' Default
167     strSlotLength = 2
168
169     ' Create RotaRequirements sheet (delete if already present)
170     On Error Resume Next
171     ' If there is already a RotaRequirements sheet then get Start/End/Slot length from it
172     Dim strParms  As String
173     strParms = Sheets("RotaRequirements" ).Range("A1" ).Text
174     ' Have we got any parms from last time?
175     If UCase(Left(strParms, 6)) = "START="  Then
176         Dim strParmList
177         strParmList = Split(strParms, Chr(10), 3)
178         strStart = Split(strParmList(0), "=" )(1)
179         strEnd = Split(strParmList(1), "=" )(1)
180         strSlotLength = Split(strParmList(2), "=" )(1)
181     End If
182     Sheets("RotaRequirements" ).Delete
183     On Error GoTo 0
184
185     Set mzRotaSheet = Sheets.Add
186     mzRotaSheet.Name = "RotaRequirements"
187
188     ' Get parameters from user
189
190     ' Start date/time
191     Dim blnFirstTime As Boolean
192     blnFirstTime = True
193     Do Until IsDate(strStart) And Not blnFirstTime
194         strStart = InputBox("What date/time do you want the rota to start? (blank to exit)" ,
            , strStart)
195         If strStart = ""  Then Exit Sub
196         blnFirstTime = False
197     Loop
198
199     ' End date/time
200     blnFirstTime = True
201     Do Until IsDate(strEnd) And Not blnFirstTime
202         strEnd = InputBox("What date/time do you want the rota to finish? (blank to exit)" ,
            , strEnd)
203         If strEnd = ""  Then Exit Sub
204         blnFirstTime = False
205     Loop
206
207     ' Slot length
208     blnFirstTime = True
209     Do Until IsNumeric(strSlotLength) And Not blnFirstTime
210         strSlotLength = InputBox("How many hours per slot?" , , strSlotLength)
1 2
```

```
1 2
        bInFirstTime = False
    Loop

    Dim dteStart
    dteStart = CDate(strStart)
    Dim dteEnd
    dteEnd = CDate(strEnd)
    Dim intSlotLength As Integer
    intSlotLength = CInt(strSlotLength)
    Dim intSlots As Integer
    intSlots = CInt(DateDiff("h" , dteStart, dteEnd) / intSlotLength)

    ' Write parms
    mzRotaSheet.Cells(1, 1).Value = "Start="  & strStart & Chr(10) _
                            & "End="  & strEnd & Chr(10) _
                            & "SlotLength="  & strSlotLength
    ' Write date headings
    Dim intSlot As Integer
    For intSlot = 1 To intSlots
        Dim dteSlotStart As Date
        dteSlotStart = DateAdd("h" , (intSlot - 1) * intSlotLength, dteStart)
        Dim dteSlotEnd As Date
        dteSlotEnd = DateAdd("h" , intSlotLength, dteSlotStart)
        Dim strDateTime As String
        strDateTime = WeekdayName(Weekday(dteSlotStart, vbSunday), True, vbSunday) _
                & " "  & Day(dteSlotStart) _
                & "/"  & Month(dteSlotStart) _
                & "/"  & Right(Year(dteSlotStart), 2) _
                & " "  & FormatDateTime(dteSlotStart, vbShortTime) _
                & " - "  & FormatDateTime(dteSlotEnd, vbShortTime)
        mzRotaSheet.Cells(1, intSlot + 2).Value = strDateTime
        mzRotaSheet.Cells(1, intSlot + 2).Orientation = 90
    Next

    ' Count rows
    zImplSheet.Range("AJ1" ).FormulaR1C1 = "=COUNTA(C[-35])"
    Dim intRowCount As Integer
    intRowCount = zImplSheet.Range("AJ1" ).Value
    zImplSheet.Range("AJ1" ).FormulaR1C1 = ""

    mintNextRow = 2
    ' Need something to keep track of what rows we're storing what
    Set mzGroupRows = New Collection

    ' Now go through the tasks
    Dim ix As Integer
    For ix = 1 To intRowCount
        ' Get start date
        Dim strPlannedStart As String
        strPlannedStart = zImplSheet.Range("E"  & ix).Value
        ' Is it a date?
        If IsDate(strPlannedStart) Then
            Dim dtePlannedStart As Date
            dtePlannedStart = Max(CDate(strPlannedStart), dteStart)
            ' Look for an end date
            Dim strPlannedEnd As String
1 2 3
```

```
1 2 3
267             strPlannedEnd = Min(zImplSheet.Range("F"  & ix).Value, dteEnd)
268             ' Is it a date?
269             If IsDate(strPlannedStart) Then
270                 Dim dtePlannedEnd As Date
271                 ' Remove a second from the end to stop going into the next slot.
272                 ' If the task ends at 2:00 we would rather have it ending at 1:59:59 so
273                 ' it doesn't go into the next (2:00-4:00) slot as well
274                 dtePlannedEnd = DateAdd("s" , -1, CDate(strPlannedEnd))
275                 ' Is it in range
276                 If dtePlannedStart < dteEnd And dtePlannedEnd > dteStart Then
277                     ' Get ID
278                     Dim strTaskID As String
279                     strTaskID = zImplSheet.Cells(ix, 1).Text
280
281                     ' Stop when we get to reversion tasks
282                     If Left(strTaskID, 2) = "5."  Then Exit For
283
284                     ' Get Title
285                     Dim strTaskTitle As String
286                     strTaskTitle = zImplSheet.Cells(ix, 2).Text
287
288                     ' Get start slot
289                     Dim intStartSlot As Integer
290                     intStartSlot = Int(DateDiff("h" , dteStart, dtePlannedStart) / intSlotLength)
                         + 1
291                     ' Get end slot
292                     Dim intEndSlot As Integer
293                     intEndSlot = Int(DateDiff("h" , dteStart, dtePlannedEnd) / intSlotLength) + 1
294
295                     ' Get Resp and Support
296                     FillSlots zImplSheet.Range("H"  & ix).Value, _
297                             "R" , intStartSlot, intEndSlot, strTaskID, strTaskTitle
298                     FillSlots zImplSheet.Range("I"  & ix).Value, _
299                             "S" , intStartSlot, intEndSlot, strTaskID, strTaskTitle
300                 End If
301             End If
302         End If
303     Next
304
305     mzRotaSheet.Select
306     Cells.Select
307     Selection.Columns.AutoFit
308     Selection.Sort Key1:=Range("A2" ), Order1:=xlAscending, _
309                 Key2:=Range("B2" ), Order2:=xlAscending, _
310                 Header:=xlYes, _
311                 OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
312     ' Resize it
313     ActiveWindow.Zoom = 75
314     ' Freeze panes
315     Range("C2" ).Select
316     ActiveWindow.FreezePanes = True
317 End Sub
```

```
318
319     Sub CreateIssue()
320         ' Only if we're on the cutover sheet
321         If UCase(ActiveSheet.Name) <> "CUTOVER"  And UCase(ActiveSheet.Name) <>
            "IMPLEMENTATION"  Then Exit Sub
322         Dim zPlanSheet As Worksheet
323         Set zPlanSheet = ActiveSheet
324
325         ' Get task row
326         Dim intTaskRow As Integer
327         intTaskRow = ActiveCell.row
328
329         ' Check it's a pucker task
330         Dim strTaskID As String
331         strTaskID = zPlanSheet.Cells(intTaskRow, 1).Value
332         ' Task ID needs to be number.number
333         If Not IsNumeric(strTaskID) Or UBound(Split(strTaskID, "." )) <> 1 Then Exit Sub
334
335         ' Make sure the issues log exists
336         If Not SheetExists("Issues Log" ) Then
337             MsgBox "'Issues Log' sheet is missing"
338             Exit Sub
339         End If
340
341         Dim zIssueSheet As Worksheet
342         Set zIssueSheet = Sheets("Issues Log" )
343
344         ' Find the next available issue
345         Dim intIssueRow As Integer
346         intIssueRow = 1
347         Do While zIssueSheet.Cells(intIssueRow, 2).Value <> ""
348             intIssueRow = intIssueRow + 1
349         Loop
350
351         ' Got it. Does it have a number?
352         Dim strIssueNum As String
353         strIssueNum = zIssueSheet.Cells(intIssueRow, 1).Value
354         Dim intIssueNum As Integer
355         If strIssueNum = ""  Then
356             intIssueNum = intIssueRow - 1
357             zIssueSheet.Cells(intIssueRow, 1).Value = intIssueNum
358         Else
359             If Not IsNumeric(strIssueNum) Then
360                 MsgBox "'Issues log' is corrupted (not in correct format). Non.numerical entry in
                    cell A"  & intIssueRow
361                 Exit Sub
362             End If
363             intIssueNum = CInt(strIssueNum)
364             If intIssueNum <> intIssueRow - 1 Then
365                 MsgBox "'Issues log' is corrupted (not in correct format). Cell A"  & intIssueRow
                    & " expected to contain "  & intIssueRow - 1
366                 Exit Sub
367             End If
368         End If
369
370         ' Looking good
    1
```

```
1
371     ' Put issue number in task
372     If zPlanSheet.Cells(intTaskRow, 3).Value = ""  Then
373         zPlanSheet.Cells(intTaskRow, 3).Value = intIssueNum
374     Else
375         zPlanSheet.Cells(intTaskRow, 3).Value = zPlanSheet.Cells(intTaskRow, 3).Value & ","
            & intIssueNum
376     End If
377     ' Put task details in Issue
378     zIssueSheet.Cells(intIssueRow, 2).Value = zPlanSheet.Cells(intTaskRow, 1).Value
379     zIssueSheet.Cells(intIssueRow, 3).Value = zPlanSheet.Cells(intTaskRow, 2).Value
380     ' Time and date
381     zIssueSheet.Cells(intIssueRow, 6).Value = Now
382     ' Status
383     zIssueSheet.Cells(intIssueRow, 12).Value = "Open"
384
385     ' Now select the issues sheet
386     zIssueSheet.Select
387     ' And select the issue description cell ready for business
388     zIssueSheet.Cells(intIssueRow, 4).Select
389
390 End Sub
391
392 Sub DeleteTask()
393     ' Delete the current task
394
395     If UCase(ActiveSheet.Name) <> "IMPLEMENTATION"  Then
396         Exit Sub
397     End If
398
399     Dim intRow As Integer
400     intRow = ActiveCell.row
401     Rows(intRow).Delete
402     If Cells(intRow, 1).HasFormula Then
403         Cells(intRow, 1).Formula = "=A"  & intRow - 1 & "+0.001"
404     End If
405 End Sub
406
407 Public Function DoesKeyExistInCollection(ByVal rstrKey As String, _
408                                          ByVal rzCollection As Collection) As Boolean
409     On Error Resume Next
410     Dim zObj
411     zObj = rzCollection(rstrKey)
412     If Err = 0 Then
413         DoesKeyExistInCollection = True
414     Else
415         DoesKeyExistInCollection = False
416     End If
417     On Error GoTo 0
418 End Function
```

```
419
420     Public Sub FillSlots(ByVal rstrGroupNameList As String, _
421                          ByVal rstrType As String, _
422                          ByVal rintStartSlot As Integer, _
423                          ByVal rintEndSlot As Integer, _
424                          ByVal rstrTaskID As String, _
425                          ByVal rstrTaskTitle As String)
426         Dim strGroupUser
427
428         ' Make sure we treat people as part of the group rather than as a group
429         rstrGroupNameList = Replace(rstrGroupNameList, Chr(10) & "-" , "\-" )
430
431         ' Now whizz round each group/user
432         For Each strGroupUser In Split(rstrGroupNameList, Chr(10))
433             ' Get rid of any spaces at the end
434             strGroupUser = Trim(strGroupUser)
435             ' Got anything to do?
436             If strGroupUser <> ""  Then
437                 ' Now split into group/user so we can validate
438                 Dim strNameBits
439                 strNameBits = Split(strGroupUser, "\-" )
440                 Dim strGroupName
441                 strGroupName = strNameBits(0)
442                 Dim strUpperGroupName As String
443                 strUpperGroupName = UCase(strGroupName)
444                 ' See if it's a proper group name or should we ignore it?
445                 If strUpperGroupName <> "N/A"  And strUpperGroupName <> "ALL"  And Right(
                    strUpperGroupName, 4) <> "ROTA"  Then
446                     ' A proper group name so on we go
447                     ' is there a user name?
448                     If UBound(strNameBits) > 0 Then
449                         Dim strUpperUserName
450                         strUpperUserName = UCase(strNameBits(1))
451                         ' Should we ignore it?
452                         If strUpperUserName = "N/A"  Or strUpperUserName = "ALL"  Or Right(
                            strUpperUserName, 4) = "ROTA"  Then
453                             ' Yes, so set the group user to be just the group.
454                             strGroupUser = strGroupName
455                         End If
456                     End If
457
458                     Dim intGroupRow As Integer
459                     intGroupRow = GetGroupRow(strGroupUser)
460                     Dim intSlot As Integer
461                     For intSlot = rintStartSlot To rintEndSlot
462                         If rstrType = "R"  Then
463                             mzRotaSheet.Cells(intGroupRow, intSlot + 2).Select
464                             With Selection.Interior
465                                 .ColorIndex = 10
466                                 .Pattern = xlSolid
467                             End With
468                         Else ' must be support
469                             mzRotaSheet.Cells(intGroupRow, intSlot + 2).Select
470                             With Selection.Interior
471                                 If .ColorIndex <> 10 Then
472                                     .ColorIndex = 46
    1 2 3 4 5 6 7 8
```

```
1 2 3 4 5 6 7 8
473             .Pattern = xlSolid
474         End If
475     End With
476 End If
477 If Selection.Comment Is Nothing Then
478     Selection.AddComment
479     Selection.Comment.Visible = False
480     Selection.Comment.Text rstrTaskID & " "  & rstrTaskTitle
481     ' Resize comment box
482     Selection.Comment.Shape.TextFrame.AutoSize = True
483 Else
484     Selection.Comment.Text Selection.Comment.Text & Chr(10) & rstrTaskID & "
        "  & rstrTaskTitle
485 End If
486 Next
487 End If
488 End If
489 Next
490
491 End Sub
```

```
492
493     Sub Format()
494
495         ' Can run formatting against Implementation or Cutover sheets
496
497         ' For the Implementation sheet it deals with fonts and background colours of the
            different
498         ' sections and section headers. It will leave any red text in the Task Title as this
499         ' is commonly used for displaying milestones
500
501         ' For the Cutover sheet it will add all the conditional formatting
502
503         If UCase(ActiveSheet.Name) <> "IMPLEMENTATION"  And UCase(ActiveSheet.Name) <>
            "CUTOVER"  Then Exit Sub
504
505         Application.ScreenUpdating = False
506
507         Range("AJ1" ).Select
508         ActiveCell.FormulaR1C1 = "=COUNTA(C[-35])"
509         Dim intRows As Integer
510         intRows = Range("AJ1" ).Value
511         ActiveCell.FormulaR1C1 = ""
512
513         Select Case UCase(ActiveSheet.Name)
514         Case "IMPLEMENTATION"
515
516             ' Calculate the total number of rows
517
518             ' Apply standard formatting
519             Dim ix As Integer
520             For ix = 1 To intRows
521                 ' Get TaskNo
522                 Dim strTaskNo As String
523                 strTaskNo = Cells(ix, 1).Value
524                 If IsNumeric(strTaskNo) Then
525                     ' Section header?
526                     If InStr(strTaskNo, "." ) < 1 Then
527                         ' No dot. must be a section header
528                         Rows(ix).Select
529                         Selection.Interior.ColorIndex = 39
530                         Selection.Font.ColorIndex = 0
531                     Else
532                         ' Now it could be that someone's already applied some formatting within the
                            row
533                         ' For example they may have highlighted part of a cell red (commonly used
                            for
534                         ' milestone highlighting)
535                         ' In this case we should only attempt to recolour if the duration cell is
                            incorrect
536                         ' which implies the row has been moved
537                         Select Case Left(strTaskNo, 1)
538                         Case "2"   ' pre-imp
539                             RecolourRow ix, 10
540                         Case "3"   ' Imp
541                             RecolourRow ix, 11
542                         Case "4"   ' Post-imp
    1 2 3 4 5 6
```

```
1 2 3 4 5 6
543             RecolourRow ix, 13
544         Case "5" , "6" , "7" , "8" , "9"  ' Reversion
545             RecolourRow ix, 46
546         Case Else
547             ' do nothing
548         End Select
549       End If
550     End If
551   Next
552
553   ' If Cutover Sheet
554 Case "CUTOVER"
555
556   For ix = 3 To intRows
557
558     Dim strRange, formatReady, formatRunning, formatFinished
559     strRange = "A"  & ix & ":Y"  & ix
560     formatReady = "=IF($V$"  & ix & ",1,0)"
561     formatRunning = "=IF($Q$"  & ix & "=" "RUNNING" ",1,0)"
562     formatFinished = "=IF($Q$"  & ix & "=" "FINISHED" ",1,0)"
563
564     Range(strRange).Select
565     Selection.FormatConditions.Delete
566
567     ' In general for each row set the following
568     'If the task is started set the back colour to ORANGE
569     Selection.FormatConditions.Add Type:=xlExpression, formula1:=formatRunning
570     Selection.FormatConditions(1).Interior.ColorIndex = 45
571
572     'If the task has finished set the back colour to GREY
573     Selection.FormatConditions.Add Type:=xlExpression, formula1:=formatFinished
574     Selection.FormatConditions(2).Interior.ColorIndex = 15
575
576     'If the task is late set the back colour to PINK
577     Selection.FormatConditions.Add Type:=xlExpression, formula1:=formatReady
578     Selection.FormatConditions(3).Interior.ColorIndex = 38
579
580     ' However, for the duration column we want to highlight tasks that are over
        duration
581     Dim formatOverdue, formatNotOverdue
582     formatReady = "=IF(AND($V$"  & ix & ",$H$"  & ix & "=0),1,0)"
583     formatOverdue = "=IF($R$"  & ix & ",1,0)"
584     formatNotOverdue = "=IF($R$"  & ix & ",0,1)"
585
586     Range("D"  & ix).Select
587     Selection.FormatConditions.Delete
588
589     'If the task is late & not started set the back colour to PINK
590     Selection.FormatConditions.Add Type:=xlExpression, formula1:=formatReady
591     Selection.FormatConditions(1).Interior.ColorIndex = 38
592
593     'If the task is overdue set the back colour to RED
594     Selection.FormatConditions.Add Type:=xlExpression, formula1:=formatOverdue
595     Selection.FormatConditions(2).Interior.ColorIndex = 3
596
597     'If the task is not overdue set the back colour to GREEN
1 2 3
```

```
1 2 3
598             Selection.FormatConditions.Add Type:=xlExpression, formula1:=formatNotOverdue
599             Selection.FormatConditions(3).Interior.ColorIndex = 43
600
601         Next ix
602         'Hide columns
603         Sheets("cutover" ).Columns("R:W" ).EntireColumn.Hidden = True
604
605     End Select
606
607     strRange = "A1:Y"  & intRows
608
609     'Put grid on all required columns
610     Range(strRange).Select
611     Selection.Borders(xlDiagonalDown).LineStyle = xlNone
612     Selection.Borders(xlDiagonalUp).LineStyle = xlNone
613     With Selection.Borders(xlEdgeLeft)
614         .LineStyle = xlContinuous
615         .Weight = xlThin
616         .ColorIndex = xlAutomatic
617     End With
618     With Selection.Borders(xlEdgeTop)
619         .LineStyle = xlContinuous
620         .Weight = xlThin
621         .ColorIndex = xlAutomatic
622     End With
623     With Selection.Borders(xlEdgeBottom)
624         .LineStyle = xlContinuous
625         .Weight = xlThin
626         .ColorIndex = xlAutomatic
627     End With
628     With Selection.Borders(xlEdgeRight)
629         .LineStyle = xlContinuous
630         .Weight = xlThin
631         .ColorIndex = xlAutomatic
632     End With
633     With Selection.Borders(xlInsideVertical)
634         .LineStyle = xlContinuous
635         .Weight = xlThin
636         .ColorIndex = xlAutomatic
637     End With
638     With Selection.Borders(xlInsideHorizontal)
639         .LineStyle = xlContinuous
640         .Weight = xlThin
641         .ColorIndex = xlAutomatic
642     End With
643
644     Application.ScreenUpdating = True
645     ActiveSheet.Range("A1:A1" ).Select
646
647 End Sub
```

```
648
649     Public Function GetGroupRow(ByVal rstrGroupName As String) As Integer
650         Dim strUpperGroupName As String
651         strUpperGroupName = UCase(rstrGroupName)
652
653         If DoesKeyExistInCollection(strUpperGroupName, mzGroupRows) Then
654             GetGroupRow = mzGroupRows(strUpperGroupName)
655         Else
656             GetGroupRow = mintNextRow
657             mintNextRow = mintNextRow + 1
658             Dim strGroupAndUser
659             strGroupAndUser = Split(rstrGroupName, "\-" )
660             mzRotaSheet.Cells(GetGroupRow, 1).Value = strGroupAndUser(0)
661             If UBound(strGroupAndUser) > 0 Then
662                 mzRotaSheet.Cells(GetGroupRow, 2).Value = strGroupAndUser(1)
663             Else
664                 mzRotaSheet.Cells(GetGroupRow, 2).Value = "-"
665             End If
666             mzGroupRows.Add GetGroupRow, strUpperGroupName
667         End If
668
669     End Function
```

```
670
671     '=============================================================================
672     '=============================================================================
673     '=============================================================================
674     '=============================================================================
675     '=============================================================================
676     ' CHECK FOR LATEST MACROS
677     '=============================================================================
678     '=============================================================================
679     '=============================================================================
680     '=============================================================================
681     '=============================================================================
682
683     Public Function GetLatestMacrosVersion() As Double
684
685         Dim zFSO
686         Set zFSO = CreateObject("scripting.filesystemobject" )
687
688         Dim zMacrosFolder
689         Set zMacrosFolder = zFSO.GetFolder(mcstrMacrosPath)
690
691         Dim zFile
692         Dim dblLatestVersion As Double
693         dblLatestVersion = 0
694         For Each zFile In zMacrosFolder.Files
695             If UCase(Left(zFile.ShortName, 16)) = "PLANAUTOMATION_V"  Then
696                 Dim strFileVersion
697                 strFileVersion = Mid(zFile.ShortName, 17, Len(zFile.ShortName) - 20)
698                 If IsNumeric(strFileVersion) Then
699                     If CDbl(strFileVersion) > dblLatestVersion Then dblLatestVersion = CDbl(
                        strFileVersion)
700                 End If
701             End If
702         Next
703
704         GetLatestMacrosVersion = dblLatestVersion
705
706     End Function
707
708     Public Function GetMacroPath() As String
709         GetMacroPath = mcstrMacrosPath
710     End Function
711
712     Public Function GetVersion() As Double
713         GetVersion = mcdblVersion
714     End Function
715
716     ' Assign to ctrl+h
717     Sub HideCompletedTasks()
718         Selection.AutoFilter Field:=17, Criteria1:="<>Finished" , Operator:=xlAnd
719     End Sub
```

```
720
721     Sub MakePlanExecutable()
722
723         Dim formula1, formula2, formula3, cell1, cell2, cell3, cell4, cell5, cell6, _
724         cell7, cell8, cell9, str, str1, str2, str3, str4, str5, str6 As String
725
726         ' Validate the status quo
727         If Not SheetExists("Implementation" ) Then
728             MsgBox "The main plan must be in a sheet called 'Implementation'"  & vbCrLf & vbCrLf
                & "Unable to continue" , vbCritical, "Error"
729             Exit Sub
730         End If
731
732         Dim zImplSheet As Worksheet
733         Set zImplSheet = Sheets("Implementation" )
734         zImplSheet.Visible = True
735
736         If Not (Trim(UCase(zImplSheet.Range("D1" ).Value)) = "DURATION" ) Then
737             zImplSheet.Range("D1" ).Select
738             MsgBox "Column D must contain duration and be titled 'Duration'"  & vbCrLf & vbCrLf &
                 "Unable to continue" , vbCritical, "Error"
739             Exit Sub
740         End If
741
742         If Not (Trim(UCase(zImplSheet.Range("E1" ).Value)) = "PLANNED START" ) Then
743             zImplSheet.Range("E1" ).Select
744             MsgBox "Column E must contain the planned start date & time and be titled 'Planned
                Start'"  & vbCrLf & vbCrLf & "Unable to continue" , vbCritical, "Error"
745             Exit Sub
746         End If
747
748         If Not (Trim(UCase(zImplSheet.Range("F1" ).Value)) = "PLANNED END" ) Then
749             zImplSheet.Range("F1" ).Select
750             MsgBox "Column F must contain the planned end date & time and be titled 'Planned
                End'"  & vbCrLf & vbCrLf & "Unable to continue" , vbCritical, "Error"
751             Exit Sub
752         End If
753
754         ' Count rows
755         zImplSheet.Range("AJ1" ).FormulaR1C1 = "=COUNTA(C[-35])"
756         Dim intRowCount As Integer
757         intRowCount = zImplSheet.Range("AJ1" ).Value
758         zImplSheet.Range("AJ1" ).FormulaR1C1 = ""
759
760         ' Check validity of plan
761         Dim ix As Integer
762         ' Go through rows, ignoring the header row
763         For ix = 2 To intRowCount
764             If Not IsNumeric(zImplSheet.Cells(ix, 1)) Then
765                 MsgBox "Non-numeric Task Id found in row "  & ix & ". Aborting"
766                 zImplSheet.Cells(ix, 1).Activate
767                 Exit Sub
768             End If
769             If zImplSheet.Cells(ix, 1) < 2 Or zImplSheet.Cells(ix, 1) > 9.999 Then
770                 MsgBox "Invalid Task Id (not in range 2 to 9) found in row "  & ix & ". Aborting"
```

1 2 3

```
1 2 3
771             zImplSheet.Cells(ix, 1).Activate
772             Exit Sub
773         End If
774
775         ' Is it a header row?
776         If InStr(zImplSheet.Cells(ix, 1), "." ) > 0 Then
777             If zImplSheet.Cells(ix, 4) = ""  Then
778                 MsgBox "Invalid Duration found in row "  & ix & ". Aborting"
779                 zImplSheet.Cells(ix, 4).Activate
780                 Exit Sub
781             End If
782             If Not IsDate(zImplSheet.Cells(ix, 5)) Then
783                 MsgBox "Invalid Planned Start found in row "  & ix & ". Aborting"
784                 zImplSheet.Cells(ix, 5).Activate
785                 Exit Sub
786             End If
787             If Not IsDate(zImplSheet.Cells(ix, 6)) Then
788                 MsgBox "Invalid Planned End found in row "  & ix & ". Aborting"
789                 zImplSheet.Cells(ix, 6).Activate
790                 Exit Sub
791             End If
792         End If
793     Next
794
795     ' Hide what we're up to
796     Application.ScreenUpdating = False
797     ' Don't want any nasty alerts
798     Application.DisplayAlerts = False
799
800     ' Create cutover sheet
801     On Error Resume Next
802     Sheets("cutover" ).Delete
803     On Error GoTo 0
804
805     Dim zCutoverSheet
806     Set zCutoverSheet = Sheets.Add
807     zCutoverSheet.Name = "Cutover"
808
809     ' And populate from implementation
810     zImplSheet.Select
811     zImplSheet.Cells.Select
812     Selection.Copy
813     zCutoverSheet.Paste
814
815     ' Insert extra columns for Actual Start/End
816     zCutoverSheet.Activate
817     zCutoverSheet.Columns("H:I" ).Select
818     Selection.Insert Shift:=xlToRight
819     ' Add column headings
820     zCutoverSheet.Range("H1" ).FormulaR1C1 = "Actual Start"
821     zCutoverSheet.Range("I1" ).FormulaR1C1 = "Actual End"
822     'Apply formats to Actual Start / End
823     zCutoverSheet.Columns("E:F" ).Copy
824     zCutoverSheet.Columns("H:I" ).PasteSpecial Paste:=xlFormats, Operation:=xlNone,
        SkipBlanks:= _
825         False, Transpose:=False
1
```

```
1
826
827         Sheets("cutover" ).Select
828         ActiveWindow.DisplayFormulas = False
829
830         'set status
831         zCutoverSheet.Range("Q3" ).FormulaR1C1 = _
832             "=if(RC[-12]=" "," "," ",IF(RC[-9]>1/1/2001,IF(RC[-8]>=RC[-9]," "Finished" ","
            "Running" "),"  "NotStarted" "))"
833         zCutoverSheet.Range("Q1" ).FormulaR1C1 = "Status"
834         zCutoverSheet.Range("Q3" ).AutoFill Destination:=Range("Q3:Q"  & intRowCount), Type:=
            xlFillDefault
835
836         zCutoverSheet.Range("R3" ).Formula = "=IF(H3=0," ""
            ",IF(I3>0,(I3-H3)>D3,(Now()-H3)>D3))"
837         zCutoverSheet.Range("R1" ).FormulaR1C1 = "Overrun?"
838         zCutoverSheet.Range("R3" ).AutoFill Destination:=Range("R3:R"  & intRowCount), Type:=
            xlFillDefault
839
840         ' Copy start column into translated start columns
841         zCutoverSheet.Columns("E:E" ).Copy
842         ' paste the above formulas into the formula worksheet
843         zCutoverSheet.Range("S1" ).PasteSpecial Paste:=xlFormulas, Operation:=xlNone, SkipBlanks
            := _
844             False, Transpose:=False
845         ' Need to preserve the INT keyword
846         zCutoverSheet.Columns("S:S" ).Replace What:="INT" , Replacement:="INZ" , LookAt:=xlPart,
             _
847             SearchOrder:=xlByColumns, MatchCase:=False
848         zCutoverSheet.Columns("S:S" ).Replace What:="T" , Replacement:="U" , LookAt:=xlPart, _
849             SearchOrder:=xlByColumns, MatchCase:=False
850         zCutoverSheet.Columns("S:S" ).Replace What:="S" , Replacement:="T" , LookAt:=xlPart, _
851             SearchOrder:=xlByColumns, MatchCase:=False
852         ' Restore the INT keyword
853         zCutoverSheet.Columns("S:S" ).Replace What:="INZ" , Replacement:="INT" , LookAt:=xlPart,
             _
854             SearchOrder:=xlByColumns, MatchCase:=False
855         zCutoverSheet.Range("S1" ).FormulaR1C1 = "Translated Start"
856
857         zCutoverSheet.Range("T3" ).Formula = "=IF(H3=0,99999,S3)"
858         zCutoverSheet.Range("T1" ).FormulaR1C1 = "Pessimistic Start"
859         zCutoverSheet.Range("T3" ).AutoFill Destination:=Range("T3:T"  & intRowCount), Type:=
            xlFillDefault
860
861         zCutoverSheet.Range("U3" ).Formula = "=IF(I3=0,99999,I3)"
862         zCutoverSheet.Range("U1" ).FormulaR1C1 = "Pessimistic End"
863         zCutoverSheet.Range("U3" ).AutoFill Destination:=Range("U3:U"  & intRowCount), Type:=
            xlFillDefault
864
865         zCutoverSheet.Range("V3" ).Formula = "=AND(S3>0,S3<=NOW())"
866         zCutoverSheet.Range("V1" ).FormulaR1C1 = "Can Start?"
867         zCutoverSheet.Range("V3" ).AutoFill Destination:=Range("V3:V"  & intRowCount), Type:=
            xlFillDefault
868
869         zCutoverSheet.Columns("S:S" ).Copy
870         zCutoverSheet.Range("W1" ).PasteSpecial Paste:=xlFormulas, Operation:=xlNone, SkipBlanks
            := _
1
```

```
1
871             False, Transpose:=False
872         zCutoverSheet.Range("W1" ).FormulaR1C1 = "Translated Start"
873
874         zCutoverSheet.Range("X3" ).Formula = "=IF(H3>0,H3,MAX(W3,NOW()))"
875         zCutoverSheet.Range("X1" ).FormulaR1C1 = "Expected Start"
876         zCutoverSheet.Range("X3" ).AutoFill Destination:=Range("X3:X"  & intRowCount), Type:=
            xlFillDefault
877
878         'zCutoverSheet.Range("Y3").Formula = "=IF(I3=0,X3+D3,I3)"
879         zCutoverSheet.Range("Y3" ).Formula = "=IF(I3=0,X3+(F3-E3),I3)"
880         zCutoverSheet.Range("Y1" ).FormulaR1C1 = "Expected End"
881         zCutoverSheet.Range("Y3" ).AutoFill Destination:=Range("Y3:Y"  & intRowCount), Type:=
            xlFillDefault
882
883         'unset the formula view
884         Sheets("cutover" ).Select
885         ActiveWindow.DisplayFormulas = False
886         zCutoverSheet.Columns("X:Y" ).NumberFormat = "@"
887
888         'Apply formats to Status & Expected Start / End
889         zCutoverSheet.Columns("E:F" ).Copy
890         zCutoverSheet.Columns("Q:Y" ).PasteSpecial Paste:=xlFormats, Operation:=xlNone,
            SkipBlanks:= _
891             False, Transpose:=False
892
893         'Apply formats to Status & Impacted Start / End
894         zCutoverSheet.Columns("E:F" ).Copy
895         zCutoverSheet.Columns("Q:Y" ).PasteSpecial Paste:=xlFormats, Operation:=xlNone,
            SkipBlanks:= _
896             False, Transpose:=False
897
898         Range("A3" ).Select
899
900         Call Format
901         'Application.ScreenUpdating = True  ' Already turned of by Format
902         Application.DisplayAlerts = True
903
904         ' Select the cutover sheet
905         zCutoverSheet.Activate
906         ' Resize it
907         ActiveWindow.Zoom = 75
908         ' Put Autofilter on
909         Selection.AutoFilter
910         ' Freeze panes
911         Range("C2" ).Select
912         ActiveWindow.FreezePanes = True
913
914         zImplSheet.Visible = xlSheetHidden
915
916         MsgBox "Done! (don't forget to save now!)" , vbOKOnly
917         Exit Sub
918
919         'ErrorInRow:
920         'MsgBox "There is something *odd* in row " & i & " which is causing an error. Please
            check it out. Aborting"
921         'Exit Sub
1
```

```
1
End Sub

'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
' COMMON UTILITY FUNCTIONS
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================

Public Function Max(r1, r2)
    If r1 > r2 Then Max = r1 Else Max = r2
End Function

Public Function Min(r1, r2)
    If r1 < r2 Then Min = r1 Else Min = r2
End Function

Sub MoveRow(ByVal rintRow As Integer, ByVal rintBeforeRow As Integer)
    Rows(rintRow).Select
    Selection.Cut
    Rows(rintBeforeRow).Select
    Selection.Insert Shift:=xlDown
End Sub
```

```
951
952     '=============================================================================
953     '=============================================================================
954     '=============================================================================
955     '=============================================================================
956     '=============================================================================
957     ' TASK MANAGEMENT
958     '=============================================================================
959     '=============================================================================
960     '=============================================================================
961     '=============================================================================
962     '=============================================================================
963
964     Sub NewTask()
965         ' Insert a task after, and dependent, on the selected one
966
967         If UCase(ActiveSheet.Name) <> "IMPLEMENTATION"  Then
968             Exit Sub
969         End If
970
971         ' Get a list of dependencies
972         Dim intDependentCells(20) As Integer
973         Dim intDependentCellIx As Integer
974         intDependentCellIx = 0
975         Dim zArea As Range
976         For Each zArea In Selection.Areas
977             Dim zRow As Range
978             For Each zRow In zArea.Rows
979                 intDependentCells(intDependentCellIx) = zRow.row
980                 intDependentCellIx = intDependentCellIx + 1
981             Next
982         Next
983
984         ' Work out where to insert the new row - after the last selected one
985         Dim intNewRow As Integer
986         ' Select last area
987         Set zArea = Selection.Areas(Selection.Areas.Count)
988         ' Get last row number and point to next row (to insert)
989         intNewRow = zArea.Rows(zArea.Rows.Count).row + 1
990         Rows(intNewRow).Select
991
992         Selection.Insert Shift:=xlDown
993         Cells(intNewRow, 1).Formula = "=A"  & intNewRow - 1 & "+0.001"
994         ' Set default time if there isn't one already!
995         If Cells(intNewRow, 4).Value = ""  Then
996             Cells(intNewRow, 4).Value = "01:00"
997         End If
998
999         If intDependentCellIx = 1 Then
1000            Cells(intNewRow, 5).Formula = "=F"  & intNewRow - 1
1001            Cells(intNewRow, 7).Formula = "=A"  & intNewRow - 1
1002        Else
1003            Dim ix As Integer
1004            Dim strDepList As String
1005            strDepList = ""
1006            Dim strStart As String
    1 2
```

```
1 2
1007        strStart = ""
1008        For ix = 0 To intDependentCellIx - 1
1009            If ix > 0 Then
1010                strDepList = strDepList & "&" ", " "&"
1011                strStart = strStart & ","
1012            End If
1013            strDepList = strDepList + "TEXT(A"  & intDependentCells(ix) & "," "0.000" ")"
1014            strStart = strStart & "F"  & intDependentCells(ix)
1015        Next
1016        Cells(intNewRow, 5).Formula = "=MAX("  & strStart & ")"
1017        Cells(intNewRow, 7).Formula = "="  & strDepList
1018    End If
1019    Cells(intNewRow, 6).Formula = "=E"  & intNewRow & "+D"  & intNewRow
1020    If Cells(intNewRow + 1, 1).HasFormula Then
1021        Cells(intNewRow + 1, 1).Formula = "=A"  & intNewRow & "+0.001"
1022    End If
1023    Rows(intNewRow).Interior.ColorIndex = xlNone
1024    Rows(intNewRow).Font.Bold = False
1025    Cells(intNewRow, 8).Font.Bold = True
1026    Cells(intNewRow, 9).Font.Bold = True
1027
1028    ' Complete the borders
1029    Dim zRange As Range
1030    Set zRange = Range("A"  & intNewRow & ":N"  & intNewRow)
1031    zRange.Borders(xlDiagonalDown).LineStyle = xlNone
1032    zRange.Borders(xlDiagonalUp).LineStyle = xlNone
1033    With zRange.Borders(xlEdgeLeft)
1034        .LineStyle = xlContinuous
1035        .Weight = xlThin
1036        .ColorIndex = xlAutomatic
1037    End With
1038    With zRange.Borders(xlEdgeTop)
1039        .LineStyle = xlContinuous
1040        .Weight = xlThin
1041        .ColorIndex = xlAutomatic
1042    End With
1043    With zRange.Borders(xlEdgeBottom)
1044        .LineStyle = xlContinuous
1045        .Weight = xlThin
1046        .ColorIndex = xlAutomatic
1047    End With
1048    With zRange.Borders(xlEdgeRight)
1049        .LineStyle = xlContinuous
1050        .Weight = xlThin
1051        .ColorIndex = xlAutomatic
1052    End With
1053    With zRange.Borders(xlInsideVertical)
1054        .LineStyle = xlContinuous
1055        .Weight = xlThin
1056        .ColorIndex = xlAutomatic
1057    End With
1058
1059    ' Ready to fill in the task
1060    Cells(intNewRow, 2).Select
1061End Sub
```

```
1062
1063    Public Sub PlanAutomation_Setup()
1064        ' Create the key shortcut for creating issues
1065        Application.MacroOptions Macro:="CreateIssue" , ShortcutKey:="i"
1066        ' and for entering date/time
1067        Application.MacroOptions Macro:="TimeUpdate" , ShortcutKey:="t"
1068        ' and for hiding completed tasks
1069        Application.MacroOptions Macro:="HideCompletedTasks" , ShortcutKey:="h"
1070        ' and for showing all tasks
1071        Application.MacroOptions Macro:="ShowAllTasks" , ShortcutKey:="a"
1072        ' and for reformatting in case that gets out of whack
1073        Application.MacroOptions Macro:="Format" , ShortcutKey:="F"
1074        ' Create the key shortcut for sorting tasks
1075        Application.MacroOptions Macro:="SortTasks" , ShortcutKey:="S"
1076        ' Add shortcut keys for adding/replacing dependencies
1077        Application.MacroOptions Macro:="AddDependency" , ShortcutKey:="q"
1078        Application.MacroOptions Macro:="ReplaceDependency" , ShortcutKey:="Q"
1079        ' Add shortcut keys for adding/deleting tasks
1080        Application.MacroOptions Macro:="NewTask" , ShortcutKey:="n"
1081        Application.MacroOptions Macro:="DeleteTask" , ShortcutKey:="D"
1082        MsgBox "Macros initialised"
1083    End Sub
1084
1085    Sub RecolourCell(ByVal rzCell As Range, ByVal rintColourIndex As Integer)
1086        ' Sets the text in a cell to the specified colour but does NOT recolour Red text
1087        Dim ix As Integer
1088        On Error GoTo MoreThan255Chars
1089        For ix = 1 To rzCell.Characters.Count
1090            With rzCell.Characters(ix, 1).Font
1091                If .ColorIndex <> 3 Then
1092                    .ColorIndex = rintColourIndex
1093                End If
1094            End With
1095        Next
1096
1097        Exit Sub
1098        MoreThan255Chars:
1099        rzCell.Font.ColorIndex = rintColourIndex
1100    End Sub
1101
1102    Sub RecolourRow(ByVal rintRow As Integer, ByVal rintColourIndex As Integer)
1103        ' Task id
1104        Cells(rintRow, 1).Font.ColorIndex = rintColourIndex
1105        ' Sort out Task Title cell (preserving red text)
1106        RecolourCell Cells(rintRow, 2), rintColourIndex
1107        ' Now the bit between Task and Task Detail
1108        Range("C"  & rintRow, "J"  & rintRow).Font.ColorIndex = rintColourIndex
1109        ' Sort out Task Detail cell (preserving red text)
1110        RecolourCell Cells(rintRow, 11), rintColourIndex
1111        ' Now the rest of the row
1112        Range("L"  & rintRow, "AZ"  & rintRow).Font.ColorIndex = rintColourIndex
1113    End Sub
```

```
1114
1115    Sub ReplaceDependency()
1116        ' Called when we want to replace a dependency
1117        ' Assign to Ctrl+Shift+Q
1118        BuildDependency (True)
1119    End Sub
1120
1121    Public Function SheetExists(sname) As Boolean
1122        '   Returns TRUE if sheet exists in the active workbook
1123        Dim x As Object
1124        On Error Resume Next
1125        Set x = ActiveWorkbook.Sheets(sname)
1126        If Err = 0 Then SheetExists = True _
1127        Else SheetExists = False
1128        On Error GoTo 0
1129    End Function
1130
1131    ' Assign to ctrl+a
1132    Sub ShowAllTasks()
1133        Selection.AutoFilter Field:=17
1134    End Sub
1135
1136    Sub ShowImplementationSheet()
1137        ' Don't want to fail just because the sheet doesn't exist
1138        On Error Resume Next
1139        Sheets("implementation" ).Visible = True
1140        On Error GoTo 0
1141    End Sub
```

```
1142
1143    Sub SortSection(ByVal rintStartRow As Integer, ByVal rintEndRow As Integer)
1144        If rintStartRow >= rintEndRow Then Exit Sub
1145
1146        ' Go through rows
1147        Dim ix As Integer
1148        For ix = rintStartRow + 1 To rintEndRow
1149            ' Is this row in the right place?
1150            ' Sort by first start time but if start times are the same then sort by end time
1151            ' so soonest finishing comes first
1152            If DateDiff("s" , Cells(ix - 1, 5), Cells(ix, 5)) < 0 _
1153                Or (DateDiff("s" , Cells(ix - 1, 5), Cells(ix, 5)) = 0 And DateDiff("s" , Cells(ix
                     - 1, 6), Cells(ix, 6)) < 0) Then
1154                ' Nope, so go find where the right position is
1155                ' Start looking at the first task in the section
1156                Dim iy As Integer
1157                iy = rintStartRow
1158                ' And keep looking through the rows until we find the right slot
1159                Do Until DateDiff("s" , Cells(ix, 5), Cells(iy, 5)) > 0 _
1160                    Or (DateDiff("s" , Cells(ix, 5), Cells(iy, 5)) = 0 And DateDiff("s" , Cells(ix,
                        6), Cells(iy, 6)) >= 0)
1161                    iy = iy + 1
1162                Loop
1163                ' And then move it
1164                MoveRow ix, iy
1165            End If
1166        Next ix
1167
1168        ' Sort out numbering
1169        ' Sort out first row
1170        Cells(rintStartRow, 1).Formula = "=A"  & rintStartRow - 1 & "+0.001"
1171        ' Then copy down
1172        Range("A"  & rintStartRow & ":A"  & rintEndRow).Select
1173        Selection.FillDown
1174
1175    End Sub
```

```
1176
1177    '=============================================================================
1178    '=============================================================================
1179    '=============================================================================
1180    '=============================================================================
1181    '=============================================================================
1182    ' TASK SORTING
1183    '=============================================================================
1184    '=============================================================================
1185    '=============================================================================
1186    '=============================================================================
1187    '=============================================================================
1188
1189    Sub SortTasks()
1190        ' Need to go through the different sections sorting them by start date
1191
1192        ' Only on Implementation sheet
1193        If UCase(ActiveSheet.Name) <> "IMPLEMENTATION"  Then
1194            MsgBox "Task sorting only available on Implementation sheet"
1195            Exit Sub
1196        End If
1197
1198        ' Warn the user to save their work
1199        If MsgBox("This macro can be destructive if your plan is not correct. You should save
            your plan before running. Do you wish to continue?" , vbYesNo) <> vbYes Then Exit Sub
1200
1201        ' Calculate the total number of rows
1202        Cells(1, 30).FormulaR1C1 = "=COUNTA(C[-29])"
1203        Dim intRows As Integer
1204        intRows = Cells(1, 30).Value
1205        Cells(1, 30).FormulaR1C1 = ""
1206
1207        Dim intSectionStart As Integer
1208        Dim ix As Integer
1209        For ix = 1 To intRows
1210            ' Get TaskNo
1211            Dim strTaskNo As String
1212            On Error Resume Next
1213            strTaskNo = Cells(ix, 1).Value
1214            If Err > 0 Then
1215                MsgBox "Invalid task id in row "  & ix & ". Aborting"
1216                Exit Sub
1217            End If
1218            On Error GoTo 0
1219            If IsNumeric(strTaskNo) Then
1220                ' Section header?
1221                If InStr(strTaskNo, "." ) < 1 Then
1222                    ' No dot. must be a section header
1223                    ' Were we already in a section?
1224                    If intSectionStart <> 0 Then
1225                        ' Yup, so we need to sort it (from section start to the previous row)
1226                        SortSection intSectionStart, ix - 1
1227                    End If
1228                    ' Next section starts with the next row
1229                    intSectionStart = ix + 1
1230                End If
    1 2 3
```

```
1  2  3
1231        End If
1232    Next
1233
1234    ' Were we in the middle of a section when we ran out of rows?
1235    If intSectionStart <> 0 Then
1236        SortSection intSectionStart, intRows
1237    End If
1238
1239    Cells(1, 1).Select
1240End Sub
```

```
1241
1242    ' Assign this macro to key combination ctrl+t
1243    Sub TimeUpdate()
1244
1245        Dim cell1 As String
1246        Dim str5 As String
1247        Dim formula1 As String
1248        Dim row As Integer
1249
1250        row = ActiveCell.row
1251
1252        Application.ScreenUpdating = False
1253
1254        Dim TimeAndDate
1255        TimeAndDate = Now
1256
1257        If UCase(ActiveSheet.Name) = "CUTOVER"  Then
1258            If ActiveCell.Column = 8 Then
1259                If Not Sheets("Cutover" ).Range("V"  & row).Value Then
1260                    If MsgBox("A dependant task has not yet completed or the time dependency has
                        not been met"  & vbCrLf & "Are you sure you want to start this task?" , vbYesNo)
                        <> vbYes Then
1261                        Exit Sub
1262                    End If
1263                End If
1264                ActiveCell.Value = TimeAndDate
1265            ElseIf ActiveCell.Column = 9 Then
1266                If Sheets("Cutover" ).Range("H"  & row).Value = 0 Then
1267                    MsgBox "Task has not yet started!"
1268                    Exit Sub
1269                End If
1270                ActiveCell.Value = TimeAndDate
1271            End If
1272            Exit Sub
1273        End If
1274
1275        If ActiveCell.Column = 9 Then
1276            ActiveCell.Value = TimeAndDate
1277        End If
1278
1279        If UCase(ActiveSheet.Name) = "ISSUES LOG"  Then
1280            If ActiveCell.Column = 6 Then
1281                ActiveCell.Value = TimeAndDate
1282            End If
1283            If ActiveCell.Column = 11 Then
1284                ActiveCell.Value = TimeAndDate
1285                ActiveSheet.Cells(ActiveCell.row, 12).Value = "Closed"
1286            End If
1287        End If
1288
1289    End Sub
1290
1291    ' Use the Upgrade sub to perform any actions that are required for these new macros
1292    Public Sub Upgrade(ByVal rdblOldVer As Double)
1293
1294    End Sub
```

```
' Version 1
'
' This macros sheet has been created to save having to use the Tools > Macro menu
'
' Change History
' V1 -  First version

Option Explicit

Private Sub cmdConstructRota_Click()
    ConstructRota
End Sub

Private Sub cmdFormat_Click()
    If SheetExists("Cutover" ) Then
        If ActiveWorkbook.Sheets("Cutover" ).Visible = xlSheetVisible Then
            ActiveWorkbook.Sheets("Cutover" ).Activate
            Format
        End If
    End If

    If SheetExists("Implementation" ) Then
        If ActiveWorkbook.Sheets("Implementation" ).Visible = xlSheetVisible Then
            ActiveWorkbook.Sheets("Implementation" ).Activate
            Format
        End If
    End If
End Sub

Private Sub cmdMakeExecutable_Click()
    If SheetExists("Implementation" ) Then
        ActiveWorkbook.Sheets("Implementation" ).Activate
        MakePlanExecutable
    End If
End Sub

Private Sub cmdShowImplementation_Click()
    ShowImplementationSheet
    ActiveWorkbook.Sheets("Implementation" ).Activate
    ActiveSheet.Range("A1:A1" ).Select
End Sub

Private Sub cmdSort_Click()
    If SheetExists("Implementation" ) Then
        ActiveWorkbook.Sheets("Implementation" ).Activate
        SortTasks
    End If
End Sub
```

```
49
50      Private Sub Worksheet_Activate()
51          If UCase(ActiveSheet.Name) = "MACROS"  Then
52              cmdSort.Enabled = False
53              cmdMakeExecutable.Enabled = False
54              cmdConstructRota.Enabled = False
55              cmdFormat.Enabled = False
56              cmdShowImplementation.Enabled = False
57
58              If SheetExists("Implementation" ) Then
59                  If ActiveWorkbook.Sheets("Implementation" ).Visible = xlSheetVisible Then
60                      cmdSort.Enabled = True
61                      cmdMakeExecutable.Enabled = True
62                      cmdFormat.Enabled = True
63                      cmdConstructRota.Enabled = True
64                  Else
65                      cmdShowImplementation.Enabled = True
66                  End If
67              End If
68
69              If SheetExists("Cutover" ) Then
70                  If ActiveWorkbook.Sheets("Cutover" ).Visible = xlSheetVisible Then
71                      cmdFormat.Enabled = True
72                  End If
73              End If
74
75          End If
76      End Sub
77
```

```
' Version 1
'
' ThisWorkbook macros sheet has been created to make the per-minute automatic refresh
' work properly. Also to allow the PlanAutomation macros to be automatically updated when
' new versions become available
'
' Change History
' V1 -  First version

Option Explicit

Private mdteRefresh As Date ' Used by AutoRefresh:

Public Sub AutoRefresh()
    mdteRefresh = Now + TimeValue("00:01:00" )
    Application.OnTime mdteRefresh, "ThisWorkbook.AutoRefresh"

    ' Go update the sheets - only if "Cutover" sheet exists
    Dim zCutoverSheet As Worksheet
    On Error Resume Next
    Set zCutoverSheet = ActiveWorkbook.Sheets("Cutover" )
    On Error GoTo 0
    If Not (zCutoverSheet Is Nothing) Then Calculate
End Sub

Function GetLatestMacrosVersion(rstrMacroPath As String) As Double

    ' Set to 0 as default
    GetLatestMacrosVersion = 0

    Dim zFSO
    Set zFSO = CreateObject("scripting.filesystemobject" )

    ' See if the upgrade path is accessible - if not then we're out of here
    If Not zFSO.FolderExists(rstrMacroPath) Then Exit Function

    Dim zMacrosFolder
    Set zMacrosFolder = zFSO.GetFolder(rstrMacroPath)

    Dim zFile
    Dim dblLatestVersion As Double
    dblLatestVersion = 0
    For Each zFile In zMacrosFolder.Files
        If UCase(Left(zFile.Name, 16)) = "PLANAUTOMATION_V"  Then
            Dim strFileVersion
            strFileVersion = Mid(zFile.Name, 17, Len(zFile.Name) - 20)
            If IsNumeric(strFileVersion) Then
                If CDbl(strFileVersion) > dblLatestVersion Then dblLatestVersion = CDbl(
                strFileVersion)
            End If
        End If
    Next

    GetLatestMacrosVersion = dblLatestVersion

1
```

```
1
End Function

'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================
' CHECK FOR LATEST MACROS
'================================================================================
'================================================================================
'================================================================================
'================================================================================
'================================================================================

Function UpgradeMacros() As Double

    Dim dblExistingMacroVersion As Double
    dblExistingMacroVersion = Application.Run("PlanAutomation.GetVersion" )

    Dim strMacroPath As String
    strMacroPath = Application.Run("PlanAutomation.GetMacroPath" )

    Dim dblLatestMacroVersion As Double
    dblLatestMacroVersion = GetLatestMacrosVersion(strMacroPath)

    ' Have we got a newer version?
    If dblLatestMacroVersion <= dblExistingMacroVersion Then Exit Function

    ' Yup, do we want to upgrade?
    If MsgBox("A new macros version (v"  & dblLatestMacroVersion & ") is available. Do you
    want to upgrade?" , vbYesNo) <> vbYes Then Exit Function

    ' Yup, so remove existing module
    ThisWorkbook.VBProject.VBComponents.Remove ThisWorkbook.VBProject.VBComponents(
    "PlanAutomation" )

    ' So, build new macro path..
    Dim strNewMacroPath As String
    strNewMacroPath = strMacroPath & "\planautomation_v"  & dblLatestMacroVersion & ".bas"

    ' And add the new one
    ThisWorkbook.VBProject.VBComponents.Import strNewMacroPath
    Application.Run "PlanAutomation.PlanAutomation_Setup"
    Application.Run "PlanAutomation.Upgrade" , dblExistingMacroVersion
End Function

' AutoRefresh stuff used to recalculate expected times and tasks overrunning etc
' Without this it requires you to change a cell to force the recalc
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    On Error Resume Next
    Application.OnTime mdteRefresh, "ThisWorkbook.AutoRefresh" , , False
    On Error GoTo 0
End Sub
```

```
106
107     Private Sub Workbook_Open()
108         UpgradeMacros ' Do we have a newer version of the macros
109         AutoRefresh ' Start the auto refreshing going
110     End Sub
```