University of Dundee

DOCTOR OF PHILOSOPHY

Use of Human Reliability Analysis to evaluate surgical technique for rectal cancer

Wilson, Peter John

Award date: 2012

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 13. Dec. 2019
Use of Human Reliability Analysis to evaluate surgical technique for rectal cancer

Peter John Wilson

2012
Appendix 5
Applescript Code for Analysis of Procedures

-- Created by Peter Wilson on 26/08/2005.
-- Copyright 2005 __MyCompanyName__. All rights reserved.
-- TaskAnalysis

-- Created by Peter Wilson on 26/08/2005.
-- Copyright (c) 2003 __MyCompanyName__. All rights reserved.
--References to lists do not need to be global (except where they originate outside the handler)
on awake from nib theObject
  global IdleTime
  set IdleTime to 1
  set ObjName to name of theObject
  if ObjName = "Control Panel" then
    SetVariables()
    CheckDirectories()
    ReadPreferences()
    DefineErrorMechanisms()
    CheckOperation()
    OpenFile()
    CheckTime()
    WriteTextBox()
  end if
end awake from nib

on idle theObject
  global s_PlayPoint, IdleTime, filename
  if IdleTime is less than 10 then
    GetMovieStats("idle")
    set contents of text field "PlayClock" of window filename to s_PlayPoint
  end if
end idle

on selected tab view item theObject tab view item tabViewItem
  global TabName
  set ObjName to name of theObject
  set TabName to name of tabViewItem
  HideBoxes(TabName)
end selected tab view item

on HideBoxes(TabName)
  global DemoOn, Cont, IOVStatus, DispY, moviecount
  set MyCol to {"Red", "Yellow", "Orange", "Green", "Label"}
  tell tab view item TabName of tab view "TabWin" of window "Control Panel"
    if TabName = "AnalysisTab" and DemoOn = false then
      repeat with c from 1 to (count of MyCol)
        set ThisField to "" & (item c of MyCol) & "Demo"
    end if
  end tell
end HideBoxes
set visible of text field ThisField to false
end repeat
if DispY is greater than 1020 then set visible of button "ShowEEM" to false
else if TabName = "PrefsTab" and (item 4 of Cont) = 0 then
    -- tell tab view item "PrefsTab"
    set MyState to contents of combo box "DeIntBox"
    set visible of text field "DeIntText" to false
    set visible of combo box "DeIntBox" to false
else if TabName = "ValidTab" and DemoOn = false then
    -- tell tab view item "ValidTab"
    repeat with c from 1 to 4
        set ThisField to "" & (item c of MyCol) & "Demo"
        set visible of text field ThisField to false
    end repeat
    set visible of text field "CurrTime" to false
    set visible of text field "CurrTimeLabel" to false
    set visible of text field "NextTime" to false
    set visible of text field "NextTimeLabel" to false
    set contents of text field "StepBox" to ""
    if IOVStatus = "IOV_Off" then
        set visible of button "NewObs" to false
        set visible of button "Accept" to false
        set visible of button "Modify" to false
        set visible of button "Reject" to false
        set visible of button "NextEpisode" to false
        -- end tell
    end if
end if
end tell
end HideBoxes

on DisplayIOV(ShowHide, IOVW)
    -- IOVWaiting = "None","No comment","Force response"
    tell tab view item "ValidTab" of tab view "TabWin" of window "Control Panel"
        set visible of button "Accept" to ShowHide
        set visible of button "Modify" to ShowHide
        set visible of button "Reject" to ShowHide
        set IOVWaiting to IOVW
    end tell
try
    if ShowHide = true then tell window "Control Panel" to set first responder
to button "Accept" of tab view item "ValidTab" of tab view "TabWin"
    end try
return IOVWaiting
end DisplayIOV

on CheckDirectories()
global myPath, ErrorPath, AnalysisPath, TemplatePath, InstrumentPath, OldDelims
    set OrigP to (path to me as string)
    set FolderList to {"ErrorDocuments", "ProceduresAnalysed", "TaskAnalysisTemplates", "Instruments"}
    set OldDelims to AppleScript's text item delimiters
    set AppleScript's text item delimiters to ":"
    --set myPath to ((text items 1 thru -5 of OrigP) & ")") as string
    set AppName to text item -2 of OrigP
    set AppleScript's text item delimiters to ".app"
    set AppName to text item 1 of AppName
    set AppleScript's text item delimiters to AppName
    set myPath to ((text item 1 of OrigP) & AppName & ":")
    set AppleScript's text item delimiters to OldDelims
    set ErrorPath to myPath & "ErrorDocuments:"
    set AnalysisPath to myPath & "ProceduresAnalysed:"
    set TemplatePath to myPath & "TaskAnalysisTemplates:"
    set InstrumentPath to myPath & "Instruments:"
    tell application "Finder"
        repeat with i from 1 to 4
            set ThisFolder to item i of FolderList as string
            set PathToFolder to (myPath & ThisFolder)
            if exists folder (PathToFolder) then
                else
                    display dialog "Folder " & ThisFolder & " not found" & return & "Making new blank folder" with icon "MyScalpel.tiff" giving up after 2
                    make new folder at (myPath as alias) with properties {name:ThisFolder}
                    end if
                end repeat
            end tell
        end CheckDirectories
    on ReadPreferences()
        global myPath, ErrorPath, AnalysisPath, TemplatePath, LastOp, Shortcuts, ErrorList, TimeSource, AbbrevList, DictRef, contentSize, ReadPrefs, SubTaskDelim, ReviewName, ReviewCol, OldDelims, SeeQT, YellowList, ReadQT, OpSelect, InstClass, PrefsSource
        set PrefFilePresent to false
        set Shortcuts to {"m", "p", "f", "e", "c", "r", "Short7"}
        set YellowList to {"Preparation", "Recovery", "Techniques"}
        set DictRef to "Dictionary"
        set contentSize to {300, 235}
        set InstClass to {"Open"}
        end ReadPreferences()
set FullInstList to {"Hand ± swab", "Diathermy coag", "Diathermy cut", "Diathermy unknown", "Knife", "Scissors", "Cautery forceps", "Babcock's / Littlewoods", "Other"}
set SubTaskDelim to {""} 
set ReviewName to "SequencesToReview"
set ReviewCol to 5
set DoNewPrefs to false
set NullPara to 0
set SeeQT to 0
set PrefsSource to 1
set ReadQT to "No (Don't show)"
tell application "Finder"
    if exists file (myPath & "Preferences") then set PrefFilePresent to true
end tell
if PrefFilePresent = true then
    set ReadPrefs to read ((myPath & "Preferences") as alias)
    set ParaCount to ((count of paragraphs in ReadPrefs) - 1)
    repeat with i from 1 to ParaCount
        set ThisLine to paragraph i of ReadPrefs
        set NextLine to paragraph (i + 1) of ReadPrefs
        if ThisLine contains "Last operation" then
            set LastOp to NextLine
        else if ThisLine contains "shortcut keys" then
            set FullErrorLine to NextLine
            set ShortcutLine to paragraph (i + 2) of ReadPrefs
            set Shortcuts to (every word in ShortcutLine)
            set ErrorList to (every word in FullErrorLine)
        else if ThisLine contains "Which of these should be highlighted" then
            set YellowList to (every word in NextLine)
        else if ThisLine contains "Name of dictionary" then
            set DictRef to word 1 of NextLine
            copy DictRef to the end of ErrorList
        else if ThisLine contains "Default Time source" then
            set TimeSource to word 1 of NextLine
        else if ThisLine contains "Marker for SubTask identification" then
            set SubTaskDelim to every word of NextLine
            if SubTaskDelim = "" or SubTaskDelim = {} then set SubTaskDelim to {""} 
        else if ThisLine contains "Word file for comments" then
            set AppleScript's text item delimiters to tab
            set ReviewName to 1st text item of NextLine
            try
                set ReviewCol to (last text item of NextLine) as integer
            on error
                set ReviewCol to 5
            end try
            set AppleScript's text item delimiters to OldDelims
        else if ThisLine contains "Show Quicktime Controls" then
            set ReadQT to NextLine
if NextLine contains "Yes" or NextLine contains "yes" then
    set SeeQT to 1
    SetButton("PrefsTab", "SeeQT", 1)
end if
else if ThisLine contains "Classification of procedure" then
    set OpTypeList to {}
    set AppleScript's text item delimiters to tab
    repeat with j from 1 to 4
        set OpLine to (paragraph (i + j) of ReadPrefs)
        if OpLine = "" then exit repeat
        set OpLine to every text item in OpLine
        copy OpLine to end of OpTypeList
    end repeat
    set AppleScript's text item delimiters to OldDelims
else if ThisLine contains "Instrument lists" then
    set InstClass to {}
    repeat with j from 1 to 4
        set InstLine to (paragraph (i + j) of ReadPrefs)
        if InstLine = "" then exit repeat
        copy InstLine to end of InstClass
    end repeat
else
    set NullPara to NullPara + 1
end if
end repeat
if NullPara = ParaCount then
display dialog "The current Preferences file is not valid" & return & 
"Would you like to overwrite it?" with icon "MyScalpel" buttons {"Yes", "No"}
default button "Yes"
    set NewPrefsButton to button returned of result
    if NewPrefsButton = "Yes" then set DoNewPrefs to true
end if
else
display dialog "Preferences file is missing" & return & "New Preferences 
file being written to Parent folder" with icon "MyScalpel" giving up after 2
    set DoNewPrefs to true
end if
if DoNewPrefs = true then WriteNewPrefs()
set OpSelect to ExtractFromList(OpTypeList)
end ReadPreferences

on WriteNewPrefs()
tell application "Finder"
    global myPath, LastOp, ReadPrefs
    set PrefFileRef to (open for access alias (myPath & "Preferences") with 
write permission)
    set ReadPrefs to "** This is the Preferences File for the Task Analysis ** 
**It must be kept in the parent folder for the program ** 
**Any changes must keep the following format: ** 
Blank Line
Last operation
Op 001

Line [x] = List of all files for error types
Line [x+1] = Corresponding shortcut keys for files

EEM Preparation  Failure  Error  Consequence  Recovery  Techniques
m p f e c r t

Which of these to be highlighted yellow when in demo (the rest will be red)?
Preparation  Recovery  Techniques

Name of dictionary file
Dictionary

Default Time source
Quicktime

Marker for SubTask identification (may use multiple, separated by tabs)
    §

Word file for comments (tab) Number of columns
VideoSeqRJCS.doc  5

Show QT Controls
No (Don't show)

Classification of procedure types (1 line for each group of classification; tab
separating members of same group)
Open  Laparoscopic
Anterior resection  APER

Instrument lists (First line indicates key word for use of Instrument1; 2nd line
for Instruemt2, etc.) 1st is default
Open
Laparoscopic

Desktop Picture
Macintosh HD:Library:Desktop Pictures:Nature:Ladybug.jpg

--End of File--"
    write ReadPrefs to PrefFileRef starting at 0
    close access PrefFileRef
    set LastOp to "Op 001"
end tell
end WriteNewPrefs
on DefineErrorMechanisms()
    global AllErrors, ErrorPath, ErrorList, ErrorListCount, Question, AbbrevList,
    DictRef, OldDelims, DictList
    set DictList to {}
    -- set AbbrevList to {}
    set AppleScript's text item delimiters to tab
    -- tell application "Finder"
    set ErrorListCount to count of items in ErrorList
    set AllErrors to {}
    repeat with i from 1 to ErrorListCount
        set ThisErrorFile to item i of ErrorList
        try
            set EEMFile to (ErrorPath & ThisErrorFile) as alias
            set EEMRead to read EEMFile
            on error
                set EEMFile to (ErrorPath & ThisErrorFile)
                try
                    close access alias EEMFile
                end try
                set EEMFileRef to (open for access alias EEMFile with write
                       permission)
                write (ThisErrorFile & return & "1. Enter your " & ThisErrorFile & "
                       types here." & return) to EEMFileRef starting at 0
                close access alias EEMFile
                set EEMRead to "* No " & ThisErrorFile & " types have been
                       defined as yet * & return & "* Use the 'Edit Error Types' button in the
                       Preferences menu to do so *"
                display dialog EEMRead with icon "MyScalpel"
            end try
            if ThisErrorFile = DictRef then
                set DictList to ListFromText(EEMRead, 2, 2, "No", ", ", 5)
            else
                set ThisErrorList to ListFromText(EEMRead, 2, 2, "No", ", ", 1)
                copy ThisErrorList to the end of AllErrors
            end if
        end try
    end repeat
    set AppleScript's text item delimiters to OldDelims
    -- set Question to {}
    repeat with i from 1 to (count of items in AllErrors)
        set ThisQuestion to TextFromList("", (item i of AllErrors), "Yes", 0, 1, 1, "")
        copy ThisQuestion to the end of Question
    end repeat
end DefineErrorMechanisms

on CheckOperation()
global OpNumber, StringLength, filename, LastOp, myPath, AnalysisPath, NewAnalysis, FileText
repeat
  ScriptToFront()
  display dialog "Last operation = " & LastOp & return & "What would you like to analyse?" default answer LastOp
  set OpToAnalyse to text returned of result
  set StringLength to (length of OpToAnalyse)
  if StringLength = 1 then
    set filename to ("00" & OpToAnalyse) as text
  else if StringLength = 2 then
    set filename to ("0" & OpToAnalyse) as text
  else
    set filename to OpToAnalyse as text
  end if
  if filename does not contain "Op" then set filename to ("Op " & filename) as text
  tell application "Finder"
    if exists file (myPath & "Finished:" & filename) then
      display dialog "The analysis of that procedure has been completed. Do you wish to Add to the analysis or Read only?" buttons ("Cancel", "Add", "Read only") default button "Read only"
        set NewAnalysis to button returned of result
      else
        set NewAnalysis to "New"
      end if
      if NewAnalysis = "Add" then set FileText to read file (myPath & "Finished:" & filename)
      end if
      if NewAnalysis = "Add" or NewAnalysis = "Read only" or NewAnalysis = "New" then exit repeat
    end if
  end tell
end repeat
if NewAnalysis = "Read only" then set AnalysisPath to myPath & "Finished:" & filename
end CheckOperation

on SetVariables()
  global s, MajTask, MinTask, subtask, newtime, errortime, MyInput, OpNumber, StringLength, filename, FilePath, ThisButton, PlayPoint, DVPlaying, ErrorPrompt, DVDPlaying, numSecs, timeStr, moviecount, CheckMainAction, List4, List4Ref, Quest4, OpType, ReadOpType, TimeSource, TaskNum, TNRef, ErrNo, LapType, TextWin, LearningMode, Shortcuts, VLCAdj, AutoSync, DemoOn, PasteSource, DVDPlayPoint,
QTRight, QTKey, AutoDemo, Cont, VLCRate, VLCPlaying,_lastTime, Inst, MyState, VLCBase, VLCData, OldMajTask, OldMinTask, OldOut2, LowNum, IOVStatus, Outline2, TabName, IOVWaiting, DispY, Sources, RealSource, Sync, QTRate, PicPath, PicNum, MatchTerms, WriteTerms

set Cont to {0, 0, 0, 0, 0}
set TabName to "AnalysisTab"
set DemoOn to false
set IOVStatus to "IOV_Off"
set DispY to 1024
HideBoxes(TabName)
set Outline2 to ""
tell window "Control Panel" to set position to {1, 1024}
tell application "TextEdit"
    launch
    close every document saving yes
end tell

try
tell application "System Events"
    tell process "Finder" to tell menu bar 1 to tell menu "Apple" to tell
    menu item "Dock" to tell menu 1 to get name of every menu item
    if result contains "Turn Hiding On" then keystroke "d" using {command
down, option down}
end tell
on error
    display dialog "Please turn on 'Access for assistive devices' in
Preferences" giving up after 3
end try
end try
set moviecount to 0
set s to 0
set VLCRate to 0.5
set VLCPlaying to 0
set MajTask to 1
set MinTask to 1
set subtask to 1
set OldMajTask to 0
set OldMinTask to 0
set OldOut2 to ""
set MyInput to 0
set PlayPoint to 0
set LowNum to 0
set DVPlaying to 0
set DVDPlaying to 0
set CheckMainAction to 0
set OpType to ""
set ReadOpType to ""
set VLCAdj to 0
set VLCBase to {1, 1, "0:0", 0, 0, 1}
set VLCData to {"VLC"}
set TimeSource to ""
set DVDCount to 0
set ReadOpTypeB to ""
set TimeElapsed to 0
set DVDPlayPoint to 180
set LapType to ""
set TextWin to "AllLarge"
set LearningMode to 0
set TaskNum to {}
set TNRef to a reference to TaskNum
repeat 20 times
  copy {} to end of TNRef
  repeat 20 times
    copy 0 to end of (item -1 of TNRef)
  end repeat
end repeat
set PasteSource to ""
set QTRight to true
set QTKey to ""
set AutoDemo to 0
set LastTime to 0
set ErrNo to 0
set QTRate to {"All", 1}
set Inst to {1, 1, "PrimInst", "SecInst"}
set IOVWaiting to "None"
set Sync to {"QT"}, {"DVD"}, {"DV"}, {"VLC"}, {"VHS"}
set Sources to {"Head1"}, {"Head2"}, {"Hitachi"}, {"Other"}
set RealSource to {"QT"}, {"DVD"}, {"DV"}, {"VLC"}, {"VHS"}
set PicPath to ""
set PicNum to 1
set MatchTerms to {"Head1", "Head 1", "Head2", "Head 2", "Hitachi", "Panasonic", "Overhead"}
  set WriteTerms to {"Head1", "Head1", "Head2", "Head2", "Hitachi", "Hitachi", "Hitachi"}
end SetVariables

on LPos(Itm, Lst, EqCon)
  set Itm to Itm as text
  repeat with i from 1 to the count of Lst
    set ThisItem to (item i of Lst)
    -- display dialog "Checking to see if" & return & "(" & Itm & ") " &
    EqCon & return & "(" & ThisItem & ")"
    if EqCon = "Contains" then
      if ThisItem contains Itm or Itm contains ThisItem then return i
    else
      if ThisItem is Itm then return i
    end if
  end repeat
  return 0
end LPos
on OpenFile()
global CurrentFile, myPath, EEMPathAndFile, AnalysisPath, ErrorPath,
TemplatePath, filename, OpRef, moviecount, Tasks, TasksRef, TAName,
NumParas, CountTA, OpType, LastLine, filename, ReadOpType,
TitlePresent, ArrayPoint, FileText, DVDCount, ReadOpTypeB, ReadSync,
TimeElapsed, HighTime, LapType, s_PlayPoint, MajTask, ErrorList,
ErrorListCount, OldDelims, ReviewName, ReviewFile, ReviewRef, OrigText,
Outputline2, EEMName, PasteWord, Inst, InstList, InstQuestion, InstView,
TrackView, OpSelect, InstClass, InstrumentPath, AllErrors, LookUpOp,
NewAnalysis, OldPic, NewPic
    set DVDCount to 0
    set ReadOpTypeB to ""
    set ReadSync to ""
    set TimeElapsed to 0
    set HighTime to 0
    set DocsNotFound to {}
    set EEMName to "ErrorModes.pdf"
    set CurrentFile to (AnalysisPath & filename) as text
    set EEMPathAndFile to (ErrorPath & EEMName) as text
    set ReviewFile to (ErrorPath & ReviewName) as text
    try
        tell application "Finder"
            set OldPic to desktop picture
            try
                set NewPic to (myPath & "Blackout.gif") as alias
                set desktop picture to file NewPic
            on error
                set NewPic to choose file with prompt "Please locate the desired
            background picture"
                set desktop picture to NewPic
            end try
            -- open file EEMPathAndFile
        end tell
    on error
        display dialog "Could not set desktop picture"
    end try
    set ArrayPoint to 0
    if NewAnalysis is not equal to "Add" then
        try
            set FileText to (read OpRef as string)
            -- set ArrayPoint to (offset of "Subtask array:" in FileText)
            on error
                display dialog "Error reading file"
set FileText to ""
end try
end if
set OrigText to FileText
if FileText contains " " then ReadLastTask()
set s_PlayPoint to CTS(HighTime)
set Tasks to {}
set TasksRef to a reference to Tasks
if OpTyp = "" then
try
  set OpType to choose from list OpSelect with prompt "Please select procedure type"
on error
    set OpType to choose from list OpSelect with prompt "Please select procedure type"
end try
  write (filename & return & "OpType: " & OpType & return) to OpRef starting at 0
else if TitlePresent = false then
  write (filename & return & "OpType: " & OpType & return) to OpRef starting at 0
end if
set LookUpOp to LPos(OpType, OpSelect, "Equals")
set InstView to 1
set TrackView to 0
set InstNo to LPos(OpType, InstClass, "Contains")
set InstFile to (InstrumentPath & "Instrument" & InstNo & ".txt") as alias
set InstRead to read InstFile
set AppleScript's text item delimiters to tab
set InstList to ListFromText(InstRead, 2, 2, "No", ",", 3)
InstWindow(1)
InstWindow(2)
-- if InstNo is greater than 1 then
--   set visible of window "Instruments" to true
set InstView to 1
set key of window "Control Panel" to true
-- end if
set InstQuestion to TextFromList("List of instruments:", InstList, "Yes", 0, 1, 1, ")
if NewAnalysis is not equal to "Read only" then write ("Date & Time of analysis: " & (current date) & return) to OpRef starting at eof
set TAName to TemplatePath & "TaskAnalysis" & LookUpOp & ".rtf"
set RTFExists to false
tell application "Finder"
  if exists file TAName then set RTFExists to true
end tell
if RTFExists is false then
display dialog "Could not find Task Analysis file" & return & "Once located, file will be saved to " & return & TemplatePath & return with icon "MyScalpel" buttons {"Browse", "Cancel", "Help"}
  set Entry to button returned of result
  if Entry = "Browse" then
    set AppleScript's text item delimiters to ":" 
tell application "Finder"
      set TAFile to choose file 
      set TANName to last text item of (TAFile as string) 
      -- display dialog filename 
      copy file TANName to folder (TemplatePath) 
      set OldName to (TemplatePath & TANName) as alias 
      set NewName to ("TaskAnalysis" & LookUpOp) 
      --& ".rtf") 
      set name of OldName to NewName 
      set TANName to (TemplatePath & NewName)
    end tell 
  else if Entry = "Help" then
    display dialog "In order to function, this program needs to be able to read from your Task Analysis File" & return & "This File should be a simple text file which contains an abbreviated analysis" buttons {"Next"} default button "Next"
    display dialog "The 1st, 11th, 21st etc lines should contain the major task headings; all the other lines will contain subtasks" buttons {"Finish"} default button "Finish"
    set TAText to ""
    set MajT to 0 
    set MinT to 0
    repeat with t from 1 to 20
      if (t + 9) mod 10 = 0 then
        set MajT to MajT + 1 
        set MinT to 0
        set TAText to TAText & ("Major Task " & MajT & ".0" & return)
      else
        set MinT to MinT + 1
        set TAText to TAText & ("Sub Task " & MajT & "." & MinT & return)
      end if
    end repeat
    set TAText to TAText & ("etc." & return) 
tell application "TextEdit"
      activate
      set NewTAFile to (make new document at beginning with
        properties {text:TAText})
      save NewTAFile in TAName 
      end tell 
  else
    CloseFile()
  end if
end if
tell application "TextEdit"
    open alias TAName
    set TAText to text of document 1
    end tell
repeat with p from 1 to (count of paragraphs in TAText)
    try
        set ParaText to (paragraph p of TAText)
        if ParaText contains "." then
            set AppleScript's text item delimiters to "."
            set ThisTask to every text item of (word 1 of ParaText)
            if (count of ThisTask) = 2 then
                set MajT to (item 1 of ThisTask) as integer
                set MinT to (item 2 of ThisTask) as integer
                Add0Task(MajT, MinT)
            end if
            set AppleScript's text item delimiters to tab
            set ParaText to (text item -1 of ParaText) as string
            --display dialog "Going to set item " & MajT & "." & MinT & " to " & ParaText buttons {"OK", "Exit") default button "OK"
            --if button returned of result = "Exit" then exit repeat
            set item (MinT + 1) of (item MajT) of TasksRef to ParaText
        end if
    end try
end repeat
try
    JumpToText(MajTask)
end try
set AppleScript's text item delimiters to OldDelims
set PasteWord to "Start"
ClickWord()
set PasteWord to "Run"
end OpenFile

on Add0Task(MajT, MinT)
global Tasks, TasksRef
repeat with m from 1 to (MajT - (count of TasksRef))
    copy {} to end of TasksRef
end repeat
repeat with m from 0 to (MinT - (count of items in (item MajT of TasksRef)))
    copy "" to end of (item MajT of TasksRef)
end repeat
end Add0Task

on ReadLastTask()
global CurrentFile, MajTask, MinTask, subtask, OutputLine, OpType, ReadOpType, HighTime, ReadSync, TimeElapsed, TitlePresent, Sync, FileText, ArrayPoint, TaskNum, TNRef, Outline2, LapType, VLCRate, CheckMainAction, OldDelims, RewoundTime, OpSelect, OpList, OpListRef, Inst

set AppleScript's text item delimiters to space
set NumPars to count of paragraphs in FileText
repeat with p from 0 to NumPars
    set ThisPara to (NumPars - p)
    set LastLine to paragraph ThisPara of FileText as string
    if LastLine is not equal to "" then
        set word1 to word 1 of LastLine
        if word1 = "TimeElapsed" and ReadSync = "" then set ReadSync to LastLine
        end if
    end repeat
    set Outputline2 to RemoveTabSpace(LastLine)
    if ReadSync is not equal to "Date" and word1 is not equal to "TimeElapsed" and LastLine contains " " then exit repeat
    end if
end repeat
set Answer to ""
set Entry to 0
set AppleScript's text item delimiters to "."
repeat with l from 1 to (count of items in OpList)
set ThisWord to item 1 of OpList
try
    set SubT to text item 3 in ThisWord as integer
    if SubT is not equal to 0 then
        set MajTask to text item 1 in ThisWord as integer
        set MinTask to text item 2 in ThisWord as integer
        set OldArray to (item (MinTask + 1) of item MajTask) of TNRef
        set NewArray to OldArray + 1
        set (item (MinTask + 1) of item MajTask) of TNRef to NewArray
    end if
end try
end repeat
try
    set MajTask to text item 1 in LastLine as integer
    set MinTask to text item 2 in LastLine as integer
end try
set AppleScript's text item delimiters to OldDelims
set TitlePresent to false
set OpType to (item 1 of OpSelect)
repeat with r from 1 to (count of items in OpSelect)
    set ThisOpType to (item r of OpSelect)
    if Txt1_2 contains ThisOpType then set OpType to ThisOpType
end repeat
if Txt1_2 contains "Op " then set TitlePresent to true
set AppleScript's text item delimiters to OldDelims
end ReadLastTask

on ReadSyncLine(TransferLine)
    global TimeElapsed, Sync, VLCRate, OldDelims, Sources, RealSource
    set AppleScript's text item delimiters to ",,"
    set LineAsList to every text item in TransferLine
    set SyncError to false
    set AppleScript's text item delimiters to "= 
    set Sync to {"QT"}, {"DVD"}, {"DV"}, {"VLC"}, {"VHS"})
    if TransferLine contains "(Source" then set Sources to {"Head1"}, {"Head2"}, {"Hitachi"}, {"Other"})
    repeat with r from 1 to (count of items in LineAsList)
        set ThisItem to (item r of LineAsList)
        set ThisSync to (last text item of ThisItem) as real
        if ThisItem contains "TimeElapsed" then set TimeElapsed to ThisSync
        if ThisItem contains "VLCRate" then set VLCRate to ThisSync
        repeat with s from 1 to (count of items in Sync)
            try
                if ThisItem contains ("Sync" & (item 1 of item s of Sync) & " ") then
                    copy ThisSync to end of (item s of Sync)
                if ThisItem contains "(Source" then
                    set ReadSource to ((word 1 of (text item -2)) of ThisItem)
                    -- set AppleScript's text item delimiters to "; ("
                    -- display dialog "(" & ReadSource & ") should be in " & (Sources as text)
set SourceIn to false
repeat with t from 1 to count of Sources
    display dialog "(" & ReadSource & ") should be in (" & (item 1 of item t of Sources) & ")"
    if (item 1 of item t of Sources) contains ReadSource then
        copy ThisSync to end of (item t of Sources)
        set SourceIn to true
    end if
end repeat
if SourceIn = false then copy ThisSync to end of (item -1 of Sources)
--set AppleScript's text item delimiters to "; ;"
--display dialog "Sources = " & (Sources as text)
--copy ReadSource to end of (item s of Sources)
--display dialog "Found a source for " & (item 1 of (item s of Sources)) & ", which is " & ReadSource
end if
end if
on error
    set SyncError to true
    display dialog "There was an error with item " & r & " which is " & ThisItem
end try
end repeat
end repeat
set AppleScript's text item delimiters to OldDelims
if SyncError = true then display dialog "Last line of Text file is incomplete: video synchronisation may not be correctly set" with icon "MyScalpel" giving up after 2
    DoTracks()
end ReadSyncLine

on Add0(MyList, MyTarget)
    set ToRep to (MyTarget - (count of items in MyList))
    repeat ToRep times
        copy 0 to end of MyList
    end repeat
    return MyList
end Add0

on StripSync(MyList)
global MatchTerms, WriteTerm
repeat with a from 1 to (count of items in MyList)
    set ThisEntry to item a of MyList
    repeat with j from 1 to (count of items in MatchTerms)
        try
            if ThisEntry contains (item j of MatchTerms) then set item a of MyList to (item j of WriteTerms)
        on error
            display dialog "Error in StripSync"
        end try
    end repeat
end repeat
end StripSync
end try
end repeat
end repeat
return MyList
end StripSync

on CheckForDVD()
global Cont, Sync, TempSync, VLCRate, moviecount, VLCPlaying,
TextWin, TimeElapsed, TimeSource, PlayPoint, MyDim, DVDDWin, DispX,
DispY, SeeQT, InstView, VLCList, MovieState, Sources, RealSource,
OldDelims, EEMPathAndFile
tell application "DVD Player"
set MyDim to info screen bounds
set DispX to item 3 of MyDim
set DispY to item 4 of MyDim
set CheckDVD to has media
set DVDDWin to viewer bounds
set controller visibility to false
end tell
--This section causes program to fail
tell window "Viewer"
set size to {1280, 1024}
set position to {DispX, DispY}
set image of image view "EEMPic" to load image "ErrorModes.pdf"
set size of image view "EEMPic" to {1280,1024}
set visible to true
end tell
if CheckDVD is true then
set (item 2 of Cont) to 1
SetButton("PrefsTab", "ControlDVD", 1)
repeat
set item 2 of RealSource to {"DVD"}
set DoDVD to "Ignore"
tell application "System Events" to tell process "DVD Player" to set
MyWin to name of every window
repeat with n from 1 to (count of items in MyWin)
if (item n of MyWin) is not equal to missing value then copy (item n of MyWin) to end of (item 2 of RealSource)
end repeat
set AppleScript's text item delimiters to "/VIDEO_TS"
set DVDDWord to text item 1 of (item -1 of (item 2 of RealSource))
set AppleScript's text item delimiters to "/"
set DVDDWord to text item -1 of DVDDWord
set AppleScript's text item delimiters to OldDelims
set ThisOp to (character -1 of DVDDWord)
set DVDDNo to {"a", "b", "c"}
set SyncOp to item ((TimeElapsed div 7150) + 1) of DVDDNo
set SyncOp to (characters 1 through -2 of DVDDWord) & SyncOp
repeat with d from 1 to count of DVDDNo
if ThisOp = (item d of DVDNo) and (TimeElapsed is not equal to ((d - 1) * 7200)) then
    ScriptToFront()
    display dialog "Time elapsed on DVD is " & TimeElapsed & " but this appears to be DVD " & d & return & "What would you like to change?"
    buttons {"DVD", "Time", "Ignore"} default button "Time"
    set DoDVD to button returned of result
    exit repeat
end if
end repeat
if DoDVD = "Time" then set TimeElapsed to (d - 1) * 7200
if DoDVD = "DVD" then
    display dialog "Please insert DVD " & SyncOp & ", then press enter"
end if
if DoDVD is not equal to "DVD" then exit repeat
else
    tell application "DVD Player"
        quit
    end tell
end if
end CheckForDVD

on CheckForQT()
    global moviucount, Cont, SeeQT, Sync, RealSource
    tell application "QuickTime Player"
        launch
        set moviucount to count of movies
        if moviucount is greater than 0 then
            set (item 1 of Cont) to 1
            if SeeQT = 0 then set controller type of every movie to none
            if SeeQT = 1 then set controller type of every movie to qtvr
        end if
        set (item 1 of RealSource) to {"QT"}
        repeat with q from 1 to moviucount
            copy name of movie q to end of (item 1 of RealSource)
        end repeat
        set item 1 of Sync to Add0(item 1 of Sync, count of items in (item 1 of RealSource))
        if (item 1 of Cont) = 1 then SetButton("PrefsTab", "ControlQT", 1)
    end tell
end CheckForQT

on CheckTime()
    global Cont, Sync, TempSync, VLRate, moviucount, VLCPlaying, TextWin, TimeElapsed, TimeSource, PlayPoint, MyDim, DispX, DispY, SeeQT, InstView, VLCLList, MovieState, Sources, RealSource, FileText, OldDelims
    set MovieState to "Paused"
set SubNo to 1
CheckForDVD()
try
    CheckForQT()
on error
    display dialog "Error in here 1a"
end try
try
    CountVLC()
on error
    display dialog "Error in here 1b"
end try
repeat with i from 1 to (count of items in Sync)
    set item i of Sync to Add0(item i of Sync, 2)
end repeat
if DispY is greater than 900 then set TextWin to "AllLarge"
if DispY is less than 901 then
    set TextWin to "QTSmall"
    tell tab view item "AnalysisTab" of tab view "TabWin" of window "Control Panel" to set visible of button "ShowEEM" to true
end if
if item 2 of (item 1 of Sync) is not equal to 0 or moviecount = 0 then set TimeSource to "DVD"
    if (item 2 of (item 2 of Sync)) is not equal to 0 or (item 2 of Cont) = 0 then set TimeSource to "Quicktime"
    if (item 1 of Cont) = 0 and (item 2 of Cont) = 0 then display dialog "Neither Quicktime movie nor DVD are present" & return & "Please remedy & restart" with icon "StopGlove.tiff" buttons {"Quit"} default button "Quit"
    CloseFile()
end if
    tell tab view item "PrefsTab" of tab view "TabWin" of window "Control Panel"
    if (item 1 of Cont) = 0 or (item 2 of Cont) = 0 then set enabled of matrix "TimeSourcePrefs" to false
    set contents of text field "PrefsElapsed" to TimeElapsed
    set state of button "Inst" to InstView
    if TimeSource = "DVD" then set current row of matrix "TimeSourcePrefs" to 2
end tell
if (count of items in (item 5 of Sync)) is greater than 1 then
    if (item 2 of (item 5 of Sync)) is not equal to 0 then set (item 5 of Cont) to 1
    SetButton("PrefsTab", "ControlVHS", 1)
end if
end if
CheckToSwap()
GetMovieStats("Startup")
set CheckQTRate to " should be played at "
if FileText contains CheckQTRate then
set AppleScript's text item delimiters to CheckQTRate
repeat with f from 1 to count of paragraphs in FileText
    set QTPar to paragraph f of FileText
    if QTPar contains CheckQTRate then exit repeat
end repeat
set QTReadName to (word -1 of text item 1 of QTPar)
set ReadQTRate to (word 1 of text item 2 of QTPar)
DoQTRate(QTReadName, ReadQTRate)
set AppleScript's text item delimiters to OldDelims
end if

PlayMovies(0)
SwapSides("Auto")
ChangeTextSize(0)
ScriptToFront()
tell window "Control Panel"
    set position to {1, 1024}
    set index to 1
    set visible to false
    set visible to true
end tell
tell window "Control Panel" to set first responder to text field "CurrEntry" of tab view item "AnalysisTab" of tab view "TabWin"
end CheckTime

on CheckToSwap()
global Sync, Sources, RealSource, SourceMovie
set AllSync to {}
repeat with a from 1 to count of Sync
    repeat with b from 2 to count of (item a of Sync)
        copy item b of item a of Sync to end of AllSync
    end repeat
end repeat
set Spares to {}
set AllSource to {}
repeat with a from 1 to count of Sources
    repeat with b from 2 to count of (item a of Sources)
        set ThisSource to item b of item a of Sources
        copy ThisSource to end of AllSource
        if AllSync does not contain ThisSource then copy ThisSource to end of Spares
    end repeat
end repeat
-- display dialog "AllSync = " & (AllSync as text) & return & "AllSource = " & (AllSource as text) & return & "Spares = " & (Spares as text)
repeat with a from 1 to count of AllSync
    set ThisSync to item a of AllSync
end repeat
if AllSource does not contain ThisSync then copy ThisSync to end of Spares
end repeat
repeat 5 times
    copy 0 to end of Spares
end repeat
--display dialog "AllSync = " & (AllSync as text) & return & "AllSource = " & (AllSource as text) & return & "Spares = " & (Spares as text)
set SpareCount to 1
repeat with a from 1 to count of RealSource
    repeat with b from 2 to count of (item a of RealSource)
        repeat with c from 1 to count of Sources
            set TS to (item 1 of item c of Sources)
            set TR to (item b of item a of RealSource)
            try
                if TR contains TS then
                    set OldSync to (item b of item a of Sync)
                    set (item b of item a of Sync) to (item 2 of item c of Sources)
                end if
            on error
                try
                    set (item b of item a of Sync) to item SpareCount of Spares
                    set SpareCount to SpareCount + 1
                on error
                    display dialog "Problem with SpareCount"
                end try
            end try
        end repeat
    end repeat
end repeat
set SourceMovie to 1
repeat with q from 2 to (count of items in (item 1 of RealSource))
    if (item q of (item 1 of Sync)) = 0 then
        set SourceMovie to (item q of (item 1 of RealSource))
        exit repeat
    end if
end repeat
DoTracks()
-- display dialog "On exit from Swap, Sync = " & (Sync as text)
return Sync
end CheckToSwap

on SetButton(DrawerOrTab, MyButton, MyState)
tell window "Control Panel"
    if DrawerOrTab is not equal to "" then
        if DrawerOrTab contains "Tab" then
            tell tab view item DrawerOrTab of tab view "TabWin"
set state of button MyButton to MyState
end tell
else
tell drawer DrawerOrTab
    set state of button MyButton to MyState
end tell
else
    set state of button MyButton to MyState
end if
end tell
end SetButton

--on MovieToFront()
-- global TimeSource
-- set MyViewer to ("" & TimeSource & " Player")
-- tell application MyViewer to activate
--end MovieToFront

on ScriptToFront()
tell application "System Events" to tell process "TaskAnalysis" to set
frontmost to true
    -- ClickMenu("TaskAnalysis", "Window", "Control Panel", "0")
end ScriptToFront

on CTS(NumSec)
    set MyDate to "1/1/1"
    set MyDec to ((NumSec mod 1) * 100) as integer
    set MyTime to (time string of ((date MyDate) + NumSec)) & "." & MyDec
end CTS

on CST(MyTime)
    set MyDec to "0."
    set AppleScript's text item delimiters to ".".
    if (count of text items of MyTime) is greater than 1 then set MyDec to MyDec & (text item 2 of MyTime)
    set NumSec to (time of (date (text item 1 of MyTime)))
    set AppleScript's text item delimiters to ""
    return (NumSec + (((MyDec as real) * 100) as integer) / 100)
end CST

on DismissBTV()
    repeat
        tell application "System Events"
            tell process "BTV Pro Carbon"
                set MyWin to name of every window
            end tell
        end tell
        if item 1 of MyWin = missing value then
display dialog "Please dismiss Registration window first"
else
    exit repeat
end if
end repeat
end DismissBTV

on PlayMovies(p)
global moviecount, Cont, TimeScale, VLCPlaying, VLCKey, MovieState, SlowButton, VLCList, QTRate
if (item 3 of Cont) = 1 then
    DismissBTV()
tell application "BTV Pro Carbon"
    play dv device
end tell
end if
if moviecount > 0 and (item 1 of Cont) = 1 then
tell application "QuickTime Player"
    set rate of every movie to 1
    if item 2 of QTRate is not equal to 1 then
        repeat with q from 1 to ((count of QTRate) div 2)
            set rate of movie (item ((q * 2) - 1) of QTRate) to (item (q * 2) of QTRate)
        end repeat
    end if
end if
if (item 4 of Cont) = 1 then
    repeat with v from 2 to (count of items in VLCList)
        VLCSpeed(1, v)
        PlayVLC("Play", MovieState, v)
    end repeat
end if
if (item 2 of Cont) = 1 then tell application "DVD Player" to play dvd
    set MovieState to "Playing"
    set SlowButton to ""
end PlayMovies

on PauseRoutine()
global Cont, moviecount, VLCPlaying, VLCKey, MovieState, VLCList, IdleTime
    set IdleTime to 10
    if MovieState contains "Speed" then
        set MovieState to "SpeedPaused"
    else if MovieState contains "Slow" then
        set MovieState to "SlowPaused"
    else
        set MovieState to "Paused"
end if
if (item 3 of Cont) = 1 then
    DismissBTV()
    tell application "BTV Pro Carbon"
        stop dv device
    end tell
end if
if (item 4 of Cont) = 1 then
    repeat with v from 2 to (count of items in VLCList)
        PlayVLC("Pause", MovieState, v)
    end repeat
end if
if (item 2 of Cont) = 1 then
    tell application "DVD Player"
        pause dvd
    end tell
end if
if moviecount > 0 and (item 1 of Cont) = 1 then
    tell application "QuickTime Player"
        stop movies
    end tell
end if
end PauseRoutine

on PlayVLC(VLCState, MovieState, v)
    global VLCList, VLCKey
    try
        tell application "System Events" to tell process (item v of VLCList) to tell
            menu bar 1 to tell menu bar item "Controls" to tell menu "Controls"
                set VLCMenuList to name of every menu item
            end tell
        if VLCMenuList contains VLCState then VLCButton(v, 6)
        on error
            display dialog "Error in activating VLCProcess " & v giving up after 2
            CountVLC()
        end try
    end PlayVLC

on VLCButton(v, VLCB)
    global VLCList
    -- Prev = 11; Next = 8; Rew = 5; Fwd = 9; Play = 6
    try
        tell application "System Events" to tell process (item v of VLCList) to tell
            window "VLC - Controller"
                click button VLCB
        end tell
    end try
end VLCButton

on VLCSpeed(NewRate, v)
global VLCKey, VLCData, VLCList
set OldRate to (item 6 of (item v of VLCData))
set (item 6 of (item v of VLCData)) to NewRate
set VLCDirection to "Null"
set VLCPress to 0
set VLCDRatio to NewRate / OldRate
if VLCDRatio is not equal to 1 then
    set VLCDirection to "Faster"
    if VLCDRatio is less than 1 then
        set VLCDRatio to (1 / VLCDRatio)
        set VLCDirection to "Slower"
    end if
    repeat with i from 1 to 16
        set VLCDRatio to VLCDRatio / 2
        if VLCDRatio = 1 then exit repeat
    end repeat
    set VLCPress to i
end if
repeat VLCPress times
    ClickMenu((item v of VLCList), "Controls", VLCDirection, "0")
delay 0.05
end repeat
end VLCSpeed

on SpeedRoutine()
global moviecount, Cont, ExitSlow, AutoSync, MyInput, ThisButton,
VLCPlaying, VLCKey, MovieState, SlowButton, NewRate, VLCList, QTRate
SyncRoutine(2)
if MyInput = "rs" or MyInput = "rv" then set MyInput to "Rew-Slow"
if SlowButton = "" then 
    PauseRoutine()
    if MyInput is not equal to "" then CapSent(MyInput)
    if ThisButton = "Speed" then
        display dialog "Do you wish to Speed through video, play Slow or Rewind then slow?" buttons {"Speed", "Slow", "Rew-Slow"} default button "Rew-Slow"
        set MyInput to button returned of result
    end if
    if MyInput = "Rew-Slow" then RewindRoutine()
    set RateDialog to "Choose new rate for movies"
    if MyInput = "Speed" then
        if SlowButton = "" then
            display dialog RateDialog buttons {"2x", "4x", "8x"} default button "2x"
            set SlowButton to button returned of result
        end if
    end if
else if MyInput = "Slow" or MyInput = "Rew-Slow" then
    display dialog RateDialog buttons {"1/2", "1/4", "1/8"} default button "1/2"
    set SlowButton to button returned of result
end if
end if
end if

set SpeedList to {"1/8", "1/4", "1/2", "2x", "4x", "8x"}
set SpeedList2 to {0.125, 0.25, 0.5, 2, 4, 8}
set NewRate to 1
repeat with i from 1 to (count of items in SpeedList)
if SlowButton = (item i of SpeedList) then set NewRate to (item i of SpeedList2)
end repeat
end if

if NewRate is less than 1 then set MovieState to "Slow"
if NewRate is greater than 1 then set MovieState to "Speed"
if (item 4 of Cont) = 1 then
repeat with v from 2 to (count of items in VLCList)
    VLCSpeed(NewRate, v)
    PlayVLC("Play", MovieState, v)
end repeat
end if
if (item 2 of Cont) = 1 then
    if MyInput = "Speed" then
        set MySpeed to " " & SlowButton & " Speed"
        ClickMenu("DVD Player", "Controls", "Scan Rate", MySpeed)
        tell application "DVD Player"
            fast forward dvd
        end tell
    else
        try
            set DVDButton to (SlowButton & " speed")
            tell application "DVD Player"
                set the_visibility to controller visibility
                if not the_visibility then set controller visibility to true
            end tell
            tell application "System Events" to tell process "DVD Player" to tell
                window 1
                tell pop up button 4 of UI element 15 of UI element 1 of UI
                    click
                    click menu item DVDButton of menu of it
                end tell
            end tell
            tell application "DVD Player" to set controller visibility to the_visibility
        end try
    end if
end if
if moviecount > 0 and (item 1 of Cont) = 1 then
    tell application "QuickTime Player"
        if not (exists movie 1) then return
        set preferred rate of every movie to NewRate
        if QTRate is not equal to {"All", 1} then
            repeat with q from 1 to (count of QTRate) div 2
try
    set preferred rate of movie (item ((q * 2) - 1) of QTRate) to
    ((item (q * 2) of QTRate) * NewRate)
end try
end repeat
end if
--MovieToFront()
end SpeedRoutine

on RewindRoutine()
    global MyInput, moviecount, TimeScale, PlayPoint, HighNum, s_PlayPoint,
    s_HighNum, s_LowNum, RewindPref, newtime, HighTime, TimeSource,
    QTPlayPoint, QTHighNum, DVDHighNum, DVDTotal, TimeElapsed, Sync,
    FileText, MajTask, MinTask, ThisButton, OldDelims, Cont, AutoSync,
    OldPlayPoint, MovieState, VLCList
    if MovieState does not contain "Paused" then PauseRoutine()
    ScriptToFront()
    -- set CheckMinus to 0
    set s_HighTime to CTS(HighTime)
    if MyInput = "rw" or MyInput = "Rew-Slow" then
        set RewindPref to "Rewind"
    else if MyInput = "st" then
        set RewindPref to "Skip to..."
    else if MyInput = "tn" then
        set RewindPref to "Task"
    else
        display dialog "Do you wish to:     Rewind," & return & "    Skip to a specific time," & return & "Or to a specific Task?"
        buttons{"Rewind", "Skip to...", "Task"} default button "Rewind"
        set RewindPref to button returned of result
    end if
    GetMovieStats("Rewind")
    -- set OldTime to PlayPoint
    if RewindPref = "Rewind" then
        display dialog "Current time point = " & s_PlayPoint & return & "Rewind by how many seconds? (+ve = Forward; -ve = Rewind)" & return & "(Last = Rewind to last entry)" buttons{"OK", "Last", "Cancel"} default button "OK"
        default answer -20
        set RewindEntry to result
        set RewindTime to text returned of RewindEntry
        set RewindButton to button returned of RewindEntry
        try
            set RewindTime to RewindTime as real
        on error
            set RewindTime to CST(RewindTime)
        end try
        if (item 2 of Cont) = 1 then
if RewindButton = "Last" then
    set RewindTime to (HighTime - PlayPoint)
    display dialog "Rewinding to " & s_HighTime buttons {"OK"} default button "OK" attached to window "Control Panel" giving up after 2
end if
set CheckMinus to ((RewindTime ^ 2) ^ 0.5)
set WindAmount to (CheckMinus / 32) + 0.06
ClickMenu("DVD Player", "Controls", "Scan Rate", "32x Speed")
-- Negative answer, so rewind
if RewindTime is not equal to 0 then
    try
tell application "DVD Player"
        activate
        if (has media is true) then
            if CheckMinus is not equal to RewindTime then
                rewind dvd
            else
                fast forward dvd
            end if
        do shell script "sleep " & (WindAmount) as string
        pause dvd
    end if
end tell
end try
end if
DVDStats()
set PlayPoint to DVDTotal + (item 2 of (item 2 of Sync))
if PlayPoint is not equal to DVDTotal then GoToNewTime()
else
    set PlayPoint to PlayPoint + RewindTime
    GoToNewTime()
end if
else if RewindPref = "Skip to..." then
    display dialog "This track = from " & s_LowNum & " to " & s_HighNum & "." & return & "Current time point = " & s_PlayPoint & ". (Default = Last Entry)" buttons {"OK", "Cancel"} default button "OK" default answer
    set s_PlayPoint to text returned of result
    try
        set OldPlayPoint to PlayPoint
        set PlayPoint to s_PlayPoint
        if s_PlayPoint contains ":" then set PlayPoint to CST(s_PlayPoint)
        GoToNewTime()
    end try
else
    set CountJump to 0
    set r to 0
    set JumpTask to (MajTask & "." & MinTask & ".0") as string
    set TasksDone to {}
    set CountJump to count of paragraphs in FileText
repeat with p from 1 to CountJump
    set ThisP to paragraph p of FileText
    if ThisP contains ".0: " and TasksDone does not contain (word 1 of ThisP) then copy word 1 of ThisP to end of TasksDone
end repeat

copy "Other" to end of TasksDone

set ChooseTask to (choose from list TasksDone with prompt "Choose which task to jump to of those recorded in this procedure") as text

if ChooseTask contains "Other" then set ChooseTask to text returned of (display dialog "Enter Task Number to jump to: " buttons {"OK", "Cancel"} default button "OK" default answer JumpTask)

    try
        set AppleScript's text item delimiters to ".
        set TaskItems to count of text items in ChooseTask
        if TaskItems = 1 then set ChooseTask to ChooseTask & 
        "0.0"
        if TaskItems = 2 then set ChooseTask to ChooseTask & 
        "0"
        set MajTask to text item 1 of ChooseTask as integer
        set MinTask to text item 2 of ChooseTask as integer
        set AppleScript's text item delimiters to OldDelims
    end try

repeat with r from 1 to CountJump
    set JumpPara to paragraph r of FileText
    if JumpPara contains ChooseTask then exit repeat
end repeat

if r = CountJump then display dialog "Task " & ChooseTask & " not encountered" buttons {"OK"} default button "OK" with icon "MyScalpel"
else
    set PlayPoint to last word of JumpPara as integer
    set PlayPoint to PlayPoint - 10
    GoToNewTime()
end if

SyncRoutine(2)
end RewindRoutine

--

on GoToNewTime()
global moviecount, PlayPoint, TimeSource, SourceMovie
if TimeSource = "DVD" then
    SyncDVDRout(3)
else
tell application "QuickTime Player"
    set CurrScale to the time scale of movie SourceMovie
    set CurrTime to current time of movie SourceMovie
    set MovTime to (PlayPoint * CurrScale)
    set the current time of movie SourceMovie to MovTime
end tell
end if
end tell
end if
end GoToNewTime

on GetMovieStats(SourceOfCall)
    global PlayPoint, HighNum, LowNum, s_PlayPoint, s_LowNum,
    s_HighNum, HighTime, TimeSource, QTPlayPoint, s_QTPlayPoint,
    QTHighNum, s_DVDTotal, DVDHighNum, TimeElapsed, OldPlayPoint,
    AutoDemo, LastTime, RewoundTime, DVDTotal, Cont
    set OldPlayPoint to PlayPoint
    if SourceOfCall = "Sync" or SourceOfCall = "Startup" then
        if TimeSource = "Quicktime" then
            DVDStats()
            QTStats()
            set PlayPoint to QTPlayPoint
            set HighNum to QTHighNum
            set LowNum to 0
            set s_PlayPoint to s_QTPlayPoint
        else
            QTStats()
            DVDStats()
            set PlayPoint to DVDTotal
            set HighNum to DVDHighNum + TimeElapsed
            set LowNum to TimeElapsed
            set s_PlayPoint to s_DVDTotal
        end if
    else
        if (item 4 of Cont) = 1 then VLCStats(PlayPoint)
            set s_LowNum to CTS(LowNum)
            set s_HighNum to CTS(HighNum)
        end if
        if PlayPoint is less than (HighTime - 10) and PlayPoint is less than
        RewoundTime then
            set AutoDemo to 1
            set RewoundTime to PlayPoint
        else
            if TimeSource = "Quicktime" then
                --DVDStats()
                QTStats()
                set PlayPoint to QTPlayPoint
            else
                DVDStats()
                set PlayPoint to DVDTotal
            end if
            set s_PlayPoint to CTS(((PlayPoint * 100) as integer) / 100)
        end if
        if PlayPoint is greater than HighTime then set AutoDemo to 0
    end if
end GetMovieStats
on DVDStats()
global Cont, LastDVDTIME, DVDDisplayPoint, DVDFrames, DVDHighNum,
HighTime, TimeElapsed, s_DVDDisplayPoint, DVDTotal, s_DVDTotal
if (item 2 of Cont) = 1 then
tell application "DVD Player"
set LastDVDTIME to DVDDisplayPoint
set DVDFrames to item 2 of DVDFrames
set DVDDisplayPoint to item 1 of DVDFrames
set DVDDisplayPoint to (((DVDDisplayPoint + (DVDFrames / 25)) * 100) as integer) / 100
set DVDHighNum to title length
set DVDTotal to (DVDDisplayPoint + TimeElapsed)
end tell
if DVDDisplayPoint is less than 60 and DVDDisplayPoint is less than (LastDVDTIME - 60) and HighTime is greater than 90 then
-- If new DVD track started (new low time, and old high time),
keeps track of Total DVD tracks; rounds up to nearest minute
display dialog "DVD Time point is lower than previously" & return & "Has a new DVD been started?" with icon "MyScalpel" buttons {"Yes", "No"}
default button "No"
set NewDisc to button returned of result
if NewDisc = "Yes" then
set TimeElapsed to HighTime
set MySecs to TimeElapsed mod 60
set AddSecs to (60 - MySecs)
set TimeElapsed to TimeElapsed + AddSecs
tell tab view item "PrefsTab" of tab view "TabWin" of window "Control Panel"
set contents of text field "PrefsElapsed" to TimeElapsed
end tell
end if
end if
set s_DVDDisplayPoint to CTS((round ((DVDDisplayPoint) * 100)) / 100)
set s_DVDTotal to CTS((round ((DVDTotal) * 100)) / 100)
end if
end DVDStats

on QTStats()
global moviecount, TimeScale, QTPlayPoint, QTHighNum, s_QTPlayPoint,
s_QTHighNum, SourceMovie
if moviecount > 0 then
tell application "QuickTime Player"
set the TimeScale to the (time scale of movie SourceMovie)
set QTPlayPoint to the ((current time of movie SourceMovie) / TimeScale)
set QTHighNum to (duration of movie SourceMovie) / TimeScale as integer
set QTPlayPoint to (round (QTPlayPoint * 100)) / 100
end tell
end if
end QTStats
end tell
set s_QTPlayPoint to CTS((round (QTPlayPoint * 100)) / 100)
set s_QTHighNum to CTS(QTHighNum)
end if
end QTStats

on CountVLC()
global OldDelims, VLCData, VLRate, Cont, VLCSyncData, VLCBase, VLCList, Sync, RealSource, Sources
tell application "System Events"
    set MyProcess to name of every process
    set VLCList to {"VLC"}
    repeat with v from 1 to (count of items in MyProcess)
        if (item v of MyProcess) = "VLC" then copy v to end of VLCList
    end repeat
end tell
try
    if VLCList is not equal to {} then
        set AddToVLCData to (count of items in VLCList) - (count of items in VLCData)
        repeat AddToVLCData times
            copy VLCBase to end of VLCData
        end repeat
        set item 4 of Sync to Add0(item 4 of Sync, count of items in VLCList)
        set AddToVLC to (count of items in VLCList) - (count of items in (item 4 of Sync))
        repeat AddToVLC times
            copy 0 to end of (item 4 of Sync)
        end repeat
        -- SetButton("PrefsTab", "ControlVLC", 1)
    end if
on error
    display dialog "Error in here 2a"
end try
end CountVLC

on VLCStats(PlayPoint)
global OldDelims, VLCData, VLRate, Cont, VLCSyncData, VLCBase, VLCList, Sync
-- VLCData is in format {Disc, VOB, "DisplayTime:00", DisplaySecs, TotalTimeAsReal, PlayRate}
    repeat with v from 2 to (count of items in VLCList)
        set (item 1 of (item v of VLCData)) to (item 1 of (DiscFromSecs(PlayPoint - (item v of (item 4 of Sync))))))
    end repeat
    tell application "VLC" to activate
    tell application "System Events"
        try
tell process (item v of VLCList) to tell menu bar 1 to tell menu bar item "Window" to tell menu "Window"
    click menu item "Controller"
end tell
on error
    display dialog "Error in VLCStats 1" giving up after 2
end try

try
    tell process (item v of VLCList) to tell window "VLC - Controller"
        set (item 3 of (item v of VLCData)) to name of static text 1
        set VLCTrack to name of static text 2
        set VLCSlider to (value of slider 1) * 1690 * VLCRate / 10000
end tell
end try
end tell

if VLCTrack = "VLC media player" then
    set VLMedia to false
    ClickMenu((item v of VLCList), "File", "Open File...", "0")
    display dialog "No media present in VLC player" & return & "Please open media then press OK" buttons {"OK", "No VLC"} default button "OK"
else
    set VLMedia to true
end if

if VLMedia = true or button returned of result = "No VLC" then exit
repeat
    end repeat
    -- display dialog "Got this far in the stats (2)" giving up after 2
    if VLMedia = true then
        set (item 4 of (item v of VLCData)) to CST(item 3 of (item v of VLCData))
        set (item 3 of (item v of VLCData)) to CTS(item 4 of (item v of VLCData))
        -- display dialog "Got this far in the stats (2b)" giving up after 2
        if ((VLCSlider - (item 4 of (item v of VLCData))) ^ 2) is less than 1 then
            set (item 4 of (item v of VLCData)) to ((VLCSlider * 100) as integer) / 100
            set AppleScript's text item delimiters to "."
            set VLCTrack to text item 1 of VLCTrack
            set AppleScript's text item delimiters to "_"
            set (item 2 of (item v of VLCData)) to (text item 3 of VLCTrack) as integer
            set AppleScript's text item delimiters to OldDelims
        set (item 5 of (item v of VLCData)) to SecsFromDisc((item v of VLCData))
        -- display dialog "Got this far in the stats (2c)" giving up after 2
        else
            set (item 4 of Cont) to 0
            try
                tell application "System Events" to tell process (item v of VLCList)
                    tell window "Open Source"
                        click button "Cancel"
end try
    end if
end if
on DiscFromSecs(VLCSecs)
    global VLCRate
    set VLCBase to {0, 0, "Time:00", 0, 0, 1}
    set VLCDiscMod to VLCSecs mod 7100
    set (item 1 of VLCBase) to (VLCSecs div 7100) + 1
    set (item 2 of VLCBase) to ((VLCDiscMod div 1690) + 1)
    set (item 4 of VLCBase) to ((VLCDiscMod mod 1690) * VLCRate)
    set (item 3 of VLCBase) to CTS(item 4 of VLCBase)
    set (item 5 of VLCBase) to VLCSecs
    return VLCBase
end DiscFromSecs

on SecsFromDisc(VLCBase)
    global VLCRate
    set VLCSecs to ((item 1 of VLCBase) - 1) * 7100
    set VLCSecs to VLCSecs + (((item 2 of VLCBase) - 1) * 1690)
    set VLCSecs to VLCSecs + ((item 4 of VLCBase) / VLCRate)
    return VLCSecs
end SecsFromDisc

on SyncVLCRout(AutoSync)
    global PlayPoint, Sync, VLCAdj, s_PlayPoint, VLCRate, VLCData, VLCList, VLCKey, VLCRate
    repeat with v from 2 to (count of items in VLCList)
        set VLCSyncTime to (PlayPoint - (item v of (item 4 of Sync)))
        set VLCSyncTime to (round (VLCSyncTime * 100)) / 100
        set VLCSyncData to DiscFromSecs(VLCSyncTime)
        -- If AutoSync = 0 = fully manual
        set VLCAnswer to "OK"
        set UseAcc to false
        if (item 4 of (item v of VLCData)) is not equal to (item 4 of (item v of VLCData)) as integer then set UseAcc to true
        if AutoSync is less than 2 then
            ScriptToFront()
            display dialog "Current VLC position is:" & tab & tab & "DVD " & (item 1 of (item v of VLCData)) & ", VOB " & (item 2 of (item v of VLCData)) & ", " & (item 3 of (item v of VLCData)) & return & "Synchronised VLC position is:" & tab & "DVD " & (item 1 of VLCSyncData) & ", VOB " & (item 2 of VLCSyncData) & ", " & (item 3 of VLCSyncData) buttons {"OK", "Use Current", "Adjust"} default button "OK" default answer (item 3 of VLCSyncData)
set VLCResult to result
set Entry to text returned of VLCResult
set VLCAnswer to button returned of VLCResult
if VLCAnswer = "Use Current" then set Entry to (item 3 of (item v of VLCData))
set (item 3 of VLCSyncData) to Entry
set (item 4 of VLCSyncData) to CST(Entry)
set (item 5 of VLCSyncData) to SecsFromDisc(VLCSyncData)
if VLCAnswer = "Adjust" then
display dialog "Enter new rate for VLC adjustment:" default answer
VLCRate
try
set VLCRate to (text returned of result) as real
display dialog "What disc number is this" default answer (item 1 of VLCSyncData)
set VLCDisc to text returned of result as integer
set (item 1 of VLCSyncData) to VLCDisc
set (item 1 of (item v of VLCData)) to VLCDisc
display dialog "What VOB do you wish to go to" default answer (item 2 of VLCSyncData)
set (item 2 of VLCSyncData) to text returned of result as integer
set (item 5 of VLCSyncData) to SecsFromDisc(VLCSyncData)
set (item 5 of (item v of VLCData)) to SecsFromDisc((item v of VLCData))
set VLCAdj to (item 5 of VLCSyncData) - (item 5 of (item v of VLCData))
end try
end if
end if
set (item v of (item 4 of Sync)) to (((PlayPoint - (item 5 of VLCSyncData)) * 100) as integer) / 100
-- Prev = 11; Next = 8; Rew = 5; Fwd = 9; Play = 6
if VLCAnswer is not equal to "Use Current" then
-- Go to correct VOB
if (item 2 of VLCSyncData) is not equal to (item 2 of (item v of VLCData)) then
set DiffInVOB to (item 2 of VLCSyncData) - (item 2 of (item v of VLCData))
set PressVOB to (DiffInVOB ^ 2) ^ 0.5 as integer
repeat PressVOB times
set PrevNext to 11
if DiffInVOB is greater than 0 then set PrevNext to 8
try
tell application "System Events"
tell process (item v of VLCList) to tell menu bar 1 to tell menu bar item "Window" to tell menu "Window"
click menu item "Controller"
end tell
tell process (item v of VLCList) to tell window "VLC -
click button PrevNext
end tell
delay 0.1
end tell
on error
display dialog "Error in SyncVLC 1" giving up after 2
end try
end repeat
end if
PlayVLC("Pause", "Paused", v)
-- Get within 20 secs of correct timepoint
set b to {"z", "x", "c"}, {"m", "n", "b"}
tell application "System Events" to tell process (item v of VLCList) to
tell window "VLC - Controller"
    set (item 3 of (item v of VLCData)) to name of static text 1
end tell
set (item 4 of (item v of VLCData)) to CST(item 3 of (item v of VLCData))
set VDiff to ((item 4 of VLCSyncData) - (item 4 of (item v of VLCData)))
if VDiff is less than 0 then
    set VList to item 1 of b
    set VDiff to VDiff - 5
else
    set VList to item 2 of b
end if
set VDiff to (VDiff ^ 2) ^ 0.5
set X to VDiff div 300
set y to (VDiff mod 300) div 60
set z to (VDiff - ((X * 300) + (y * 60))) div 10
set c to {X, y, z}
tell application "VLC" to activate
repeat with RptVLC from 1 to 3
    set ThisV to item RptVLC of VList
    repeat (item RptVLC of c) times
        tell application "System Events" to tell process "VLC"
            keystroke ThisV
        end tell
        delay 0.1
    end repeat
end repeat

-- Fast forward to within 1 sec of correct timepoint
VLCSpeed(4, v)
PlayVLC("Play", "Paused", v)
repeat
    if UseAcc = false then
        tell application "System Events" to tell process (item v of VLCList) to tell window "VLC - Controller"
            set (item 3 of (item v of VLCData)) to name of static text 1
end tell
set (item 4 of (item v of VLCData)) to CST(item 3 of (item v of VLCData))
else
tell application "System Events" to tell process (item v of VLCList) to tell window "VLC - Controller"
set (item 4 of (item v of VLCData)) to (value of slider 1) * 1690 / 10000
end tell
end if
if (item 4 of (item v of VLCData)) is greater than ((item 4 of VLCSyncData) - 3) then exit repeat
end repeat
PlayVLC("Pause", "Speed", v)
VLCSpeed(1, v)
-- set MyDate to current date
-- set newtime to time of MyDate
-- display dialog "That took " & (newtime - OldTime) & " seconds"
giving up after 3
end if
end repeat
-- Move other applications to new VLC timepoint
VLCStats(PlayPoint)
set VLCTotal to SecsFromDisc((item 2 of VLCData))
if VLCAnswer is not equal to "Use Current" then
set PlayPoint to (item 2 of (item 4 of Sync)) + VLCTotal
set PlayPoint to (((PlayPoint * 100) as integer) / 100)
GoToNewTime()
end if
ScriptToFront()
end SyncVLCRout

on SyncDVRout(AutoSync)
global PlayPoint, Sync
set BTVSyncTime to PlayPoint + (item 2 of (item 3 of Sync))
set BTV to GetBTV()
if AutoSync = 0 then
set s_BTV to CTS(BTV)
set s_BTVSyncTime to CTS(BTVSyncTime)
display dialog "Current time of DV tape is " & s_BTV & return & "Sync time for DV tape is " & s_BTVSyncTime default answer s_BTVSyncTime
set s_BTVSyncTime to text returned of result
set BTVSyncTime to CST(s_BTVSyncTime)
set (item 2 of (item 3 of Sync)) to (((BTVSyncTime - PlayPoint) * 100) as integer) / 100
end if
repeat
set BTVDiff to BTVSyncTime - BTV
if BTVDiff ^ 2 is less than 25 then exit repeat
if BTVDiff is less than -180 then
set ThisGap to -90
else if BTVDiff is greater than 180 then
    set ThisGap to 90
else
    set ThisGap to BTVDiff / 2
end if
if ThisGap is less than 8.5 and ThisGap is greater than 0 then set ThisGap to 8.5
if ThisGap is greater than -8.5 and ThisGap is less than 0 then set ThisGap to -8.5
-- if (BTVDiff^2) is less than 225 then WindTape({"p","p"})
if BTV is less than BTVSyncTime then
    WindTape({"f"})
else if BTV is greater than BTVSyncTime then
    WindTape({"r"})
end if
repeat
    set BTV to GetBTV()
    if BTV is greater than (BTVSyncTime - ThisGap) and BTVDiff is greater than 0 then
        WindTape({"s"})
        exit repeat
    else if BTV is less than (BTVSyncTime - ThisGap) and BTVDiff is less than 0 then
        WindTape({"s"})
        exit repeat
    end if
end repeat
delay 2
set BTV to GetBTV()
end repeat
if BTV is less than BTVSyncTime then WindTape({"p"})
repeat
    if BTV is greater than (BTVSyncTime - 1) then
        WindTape({"s"})
        exit repeat
    end if
    set BTV to GetBTV()
end repeat
end SyncDVRout

on SyncDVRout(AutoSync)
global PlayPoint, Sync, DVDTot, TimeElapsed, s_DVDPlayPoint, RealSource
tell application "DVD Player" to set TitleLength to title length
set SyncDVDPlayPoint to (PlayPoint - (item 2 of (item 2 of Sync))) - TimeElapsed
set SyncDVDPlayPoint to ((SyncDVDPlayPoint * 100) as integer) / 100
set DT to (PlayPoint - (item 2 of (item 2 of Sync)))/ 100
end SyncDVRout
set s_DT to CTS(DT)
if AutoSync = 0 then
    set s_SyncDVD to CTS(SyncDVDPlayPoint)
    display dialog "Current DVD playpoint = " & s_DVDPlayPoint & return &
    "Synchronised DVD time = " & s_SyncDVD buttons {"OK", "Current", 
    "Cancel") default button "OK" default answer s_SyncDVD
    set Entry to result
    if button returned of Entry = "Current" then
        set s_SyncDVD to s_DVDPlayPoint
    else
        set s_SyncDVD to text returned of Entry
    end if
    set DVDPlayPoint to CST(s_SyncDVD)
    set (item 2 of (item 2 of Sync)) to (PlayPoint - (DVDPlayPoint +
    TimeElapsed))
else
    end if
    if DVDPlayPoint is less than 0 then
        if DT is greater than 0 then display dialog "" & s_DT & " is on a previous
        DVD. This DVD covers timepoints from " & CTS(TimeElapsed) & 
        " to " &
        CTS(TitleLength + TimeElapsed) & return & "Please change DVD and enter
        new time in Prefs" with icon "MyScalpel"
        else if DVDPlayPoint is greater than TitleLength then
            display dialog "" & s_DT & " is on a subsequent DVD. This DVD covers
            timepoints from " & CTS(TimeElapsed) & 
            " to " & CTS(TitleLength +
            TimeElapsed) & return & "Please change DVD and enter " & (TimeElapsed +
            TitleLength) & 
            " as Time Elapsed in Prefs" with icon "MyScalpel"
            else
                set DVDDisc to "Continue"
                repeat with d from 2 to (count of (item 2 of RealSource))
                    set e to (item d of (item 2 of RealSource))
                    if e contains "VOLUME IDENTIFIER" then
                        display dialog "Warning: DVD is on a disc, and there may be a
                        delay" & return & "Continue or change time of other movies?" buttons
                        {"Continue", "Change") default button "Change"
                        set DVDDisc to button returned of result
                    end if
                end repeat
                if DVDDisc = "Continue" then
                    tell application "DVD Player"
                        set elapsed time to DVDPlayPoint
                        pause dvd
                    end tell
                    set DVDTotal to DVDPlayPoint + TimeElapsed
                else
                    set PlayPoint to (DVDPlayPoint + TimeElapsed + (item 2 of (item 2 of
                    Sync)))
                    GoToNewTime()
                end if
            end if
        end if
    end if
on DoQTRate(QTReadName, NewRate)
    global moviecount, QTRate, OldDelims
    set QTRate to {}
    set QTFixRate to 1
    set QTFixName to "None"
    tell application "QuickTime Player"
        activate
        if QTReadName is not equal to "None" then display dialog "Rate adjustment detected from text file" giving up after 2
            repeat with q from 1 to moviecount
                set QTName to name of movie q
                set DefRate to 1
                if QTName contains QTReadName then set DefRate to NewRate
                display dialog "Enter new rate for movie " & q & ", " & QTName & "" default answer DefRate
                set QTR to (text returned of result) as real
                if QTR is not equal to 1 then
                    set AppleScript's text item delimiters to "."
                    set QTFixRate to QTR
                    set QTFixName to (text item 1 of QTName)
                    set AppleScript's text item delimiters to OldDelims
                end if
                copy QTName to end of QTRate
                copy QTR to end of QTRate
            end repeat
        end if
    end tell
    if NewRate = 1 then
        set RateLine to "NB: *** " & QTFixName & " should be played at " & QTFixRate & " rate ***"
        set contents of text field "CurrEntry" of tab view item "AnalysisTab" of tab view "TabWin" of window "Control Panel" to RateLine
        display dialog "Please place the default text into the analysis file" giving up after 2
    end if
end DoQTRate

on SyncQTRout(AutoSync)
    global moviecount, PlayPoint, s_PlayPoint, TimeScale, Cont, DVDTotal, Sync, RealSource, TimeSource, QTPlayPoint, QTRate
    if item 2 of (item 1 of Sync) = 0 and TimeSource = "DVD" then set item 2 of (item 1 of Sync) to (round ((DVDTotal - QTPlayPoint) * 100)) / 100
    repeat with q from 2 to moviecount + 1
        try
            set QTR to 1
            if (item q of (item 1 of Sync)) is not equal to 0 then
                set ThisMovie to (item q of (item 1 of RealSource))
            end if
        end try
    end repeat
end SyncQTRout
tell application "QuickTime Player"
    set CurrScale to the time scale of movie ThisMovie
    set CurrTime to current time of movie ThisMovie
end tell

set ThisQT to (((PlayPoint - (item q of (item 1 of Sync))) * 100) as integer) / 100

if QTRate contains ThisMovie then
    repeat with r from 1 to (count of QTRate) - 1
        set QTRateName to item r of QTRate
        if QTRateName = ThisMovie then
            set QTR to item (r + 1) of QTRate
            set ThisQT to ThisQT * QTR
            exit repeat
        end if
    end repeat
end if

if AutoSync = 0 then
    set s_Cur to CTS(CurrTime / CurrScale)
    set s_SyncQT to CTS(ThisQT)
    display dialog "" & s_PlayPoint & " = Playpoint. Movie " & ThisMovie & " is at: " & return & s_Cur & ". Synchronised time is:" buttons {"OK", "Current", "Cancel"} default button "OK" default answer s_SyncQT
    set Entry to result
    if button returned of result = "Current" then
        set s_SyncQT to s_Cur
    else
        set s_SyncQT to text returned of Entry
    end if
    set ThisQT to CST(s_SyncQT)
    set AdjQT to ThisQT / QTR
    set ThisSync to (((PlayPoint - AdjQT) * 100) as integer) / 100
    if ThisSync = 0 then set ThisSync to 0.1
    set (item q of (item 1 of Sync)) to ThisSync
end if

set MovTime to (ThisQT * CurrScale)
tell application "QuickTime Player" to set the current time of movie ThisMovie to MovTime
end if
on error
    display dialog "Could not set time of movie " & ThisMovie
end try
end repeat
end SyncQTRout

on SyncRoutine(AutoSync)
    global moviecount, TimeScale, PlayPoint, HighNum, s_PlayPoint, HighTime, TimeSource, Cont, QTPlayPoint, s_QTPlayPoint, QTHighNum, DVDDPlayPoint, DVDDHighNum, Sync, TimeElapsed, DVDDPlaying, s_DVDDPlayPoint, DecimalFromString, DVDFrames, VLCRate, VLCAdj, OldPlayPoint, MovieState, DVDTotal
-- AutoSync: 0 = Prompts for every source; 1 = Pauses; 2 = Adjusts VLC, QT & DVD; 3 = Adjusts only QT & DVD
-- NB Sync times that are exported are times to be added on to achieve synchronisation, i.e. -ve means the relevant track is ahead of the reference track
-- if AutoSync is less than 2 and
if MovieState does not contain "Paused" then PauseRoutine()
GetMovieStats("Sync")
if AutoSync is less than 3 then
  if (item 4 of Cont) = 1 then SyncVLCRout(AutoSync)
  if (item 3 of Cont) = 1 then SyncDVRout(AutoSync)
end if
if (item 2 of Cont) = 1 and TimeSource is not equal to "DVD" then
SyncDVDRout(AutoSync)
if (item 1 of Cont) = 1 then
  if moviecount is greater than 1 or TimeSource is not equal to "Quicktime" then
    SyncQTRout(AutoSync)
  end if
if (item 5 of Cont) = 1 and AutoSync is less than 3 then
  set s_VHSTime to (PlayPoint - (item 2 of (item 5 of Sync)))
  set s_VHSTime to CTS(VHSTime)
  display dialog "Current time = " & s_VHSTime & return & "Enter synchronised time for VHS tape: " default answer s_VHSTime
  set s_VHSTime to CST(s_VHSTime)
  set (item 2 of (item 5 of Sync)) to (((PlayPoint - VHSTime) * 100) as integer) / 100
  end if
set s_PlayPoint to CTS(((PlayPoint * 100) as integer) / 100)
end SyncRoutine

on GetBTV()
global OldDelims
tell application "BTV Pro Carbon" to set BTVList to (get dtape time)
set AppleScript's text item delimiters to ":"
set s_BTV to BTVList as text
set BTV to CST(s_BTV)
set AppleScript's text item delimiters to OldDelims
return BTV
end GetBTV

on WindTape(Instructions)
tell application "BTV Pro Carbon"
  activate
tell application "System Events"
    repeat with a from 1 to count of items in Instructions
      keystroke (item a of Instructions)
      if a is less than (count of items in Instructions) then delay 1
    end repeat
end tell
end tell
end WindTape

on CollectData()
global ThisButton, Entry, MyInput, OpRef, filename, OutputLine, RewindPref, moviecount, MajTask, MinTask, Tasks, TasksRef, Cont, ExitSlow, DemoOn, ToPaste, OldDelims, Outputline2, MyRoutine, AutoDemo, MovieState, SlowButton, NewRate
set Entry to 0
set DoEnd to false
if ThisButton = "Instruments" then set MyInput to "inst"
if AutoDemo = 1 then GetMovieStats("CollectData")
if ThisButton = "PauseButton" or MyInput = "." then
  PauseRoutine()
  if MovieState does not contain "Manual" then set MovieState to MovieState & "Manual"
else if ThisButton = "Rewind" or MyInput = "rw" or MyInput = "st" or MyInput = "tn" then
  RewindRoutine()
else if ThisButton = "SyncButton" or MyInput = "sync" then
  SyncRoutine(0)
  DoTracks()
else if ThisButton = "Speed" or MyInput = "slow" or MyInput = "speed" or MyInput = "rs" or MyInput = "rv" then
  set SlowButton to ""
  SpeedRoutine()
else if ThisButton = "EndButton" or MyInput = "end" then
  set DoEnd to true
else if ThisButton = "DemoMode" or MyInput = "demo" then
  ToggleDemo()
  DemoMode(0)
else if MyInput = "word" then
  ClickWord()
  set MovieState to ReplaceChars(MovieState, "Manual", "")
else if AutoDemo = 1 and DemoOn = false then
  display dialog "Movies are at a lower timepoint than last entry" & return & "Start Demo mode? (Will start demo in 3 secs)" with icon "MyScalpel"
bUTTONS {"OK", "No"} default button "No" giving up after 3
  set AutoButton to button returned of result
  set MovieState to ReplaceChars(MovieState, "Manual", "")
  if AutoButton = "" then set AutoButton to "OK"
  set AutoDemo to 0
  if AutoButton = "OK" then
    set DemoOn to true
    DemoMode(0)
  end if
else if (ThisButton = "PlayButton" or ThisButton = "ForcePlay") and MyInput = "" then
  if MovieState = "Speed" or MovieState = "Slow" then set MovieState to "Paused"
set MovieState to ReplaceChars(MovieState, "Manual", "))
else if DemoOn = true then
display dialog "Input disabled during demo mode" attached to window "Control Panel" with icon "MyScalpel" giving up after 2
set contents of text field "CurrEntry" of tab view item "AnalysisTab" of tab view "TabWin" of window "Control Panel" to ""
else if MyInput = "edit" then
   EditLast()
else if MyInput is not equal to "" then
   OperativeStep()
   set MovieState to ReplaceChars(MovieState, "Manual", "")
end if
if DoEnd = true then
   EndRoutine()
else
   CheckToPlay()
   ScriptToFront()
end if
end CollectData

on CheckToPlay()
global MovieState, MyInput, IOVWaiting, ThisButton, IdleTime
if IOVWaiting = "Force response" and ThisButton is not equal to "ForcePlay" then
display dialog "Cannot proceed without Inter-Observer Validation" giving up after 2
else
   if MovieState = "SpeedPaused" then
      set MyInput to "Speed"
      SpeedRoutine()
   else if MovieState = "SlowPaused" then
      set MyInput to "Slow"
      SpeedRoutine()
   else if MovieState = "Paused" then
      PlayMovies(2)
   end if
   if IdleTime is not equal to 1 and MovieState does not contain "Paused" then
      set IdleTime to 1
   end if
end if
end CheckToPlay

on OperativeStep()
global s, MajTask, MinTask, subtask, MyInput, OpNumber, StringLength, filename, FilePath, OpRef, OutputLine, ErrorPrompt, CheckMainAction, ErrNo, NoEntryYet, PlayPoint, MinTaskCheck, WordLength, Shortcuts, Tasks, TasksRef, SubTaskDelim
try
   set CheckMainAction to MyInput as integer
set TotalTasks to (count of TasksRef)
if CheckMainAction is greater than (TotalTasks + 1) then
    repeat
        display dialog "Are you certain that you wish to jump ahead to Task " & CheckMainAction & "?" & return & "There are only " & TotalTasks & " entered so far." & return & "Enter next task number:" with icon "MyScalpel"
        default answer (TotalTasks + 1)
        set NewAction to text returned of result
        try
            set NewAction to NewAction as integer
            exit repeat
        on error
            display dialog "Please enter an integer" with icon "MyScalpel"
giving up after 2
        end try
    end repeat
    set CheckMainAction to NewAction
end if

GetMovieStats("OpStep")
set subtask to ReadSubTask()
set MinTaskCheck to false
set NoEntryYet to true
set OriginalInput to MyInput
set StringLength to length of MyInput
set char1 to first character of MyInput
set LastChar to last character of MyInput
set NumWords to number of words of MyInput
if CheckMainAction is greater than or equal to 1 then
    set MajTask to CheckMainAction
    set MinTask to 0
    LookUpTask()
    JumpToText(MajTask)
    ScriptToFront()
    set CheckMainAction to 0
    --Subtasks
else if MyInput = "inst" then
    ErrorSafe("inst", 0, "")
else if SubTaskDelim contains LastChar then
    set MinTaskCheck to true
    set TaskNumber to characters 1 through (StringLength - 1) of MyInput
    try
        set MinTask to TaskNumber as string
        set MinTask to MinTask as integer
        -- set subtask to 0
        LookUpTask()
    end try
else if MyInput = "d" then
    set MyInput to "Delay"
set subtask to subtask + 1
AddTaskNumbersToOutput()
else if MyInput = "nv" then
    set MyInput to "No view"
    set subtask to subtask + 1
    AddTaskNumbersToOutput()
else if MyInput = "dv" then
    set MyInput to "Diathermy via instrument"
    set subtask to subtask + 1
    AddTaskNumbersToOutput()
else
    repeat with i from 1 to NumWords
        set ErrNo to 0
        set SubErr to ""
        set CW to word i of OriginalInput
        set WordLength to length of CW
        if WordLength is less than 5 then
            if CW = "snap" then
                display dialog "Take a photo?"
                if button returned of result = "OK" then DoSnap("TakeSnap")
            end if
            repeat with s from 1 to (count of Shortcuts)
                if CW begins with (item s of Shortcuts) then
                    set StartChar to (length of (item s of Shortcuts)) + 1
                    repeat with e from StartChar to WordLength
                        set ThisChar to (character e of CW)
                        try
                            set ErrNo to ErrNo * 10 + (ThisChar as integer)
                        on error
                            if e = StartChar then exit repeat
                            set SubErr to SubErr & ThisChar
                        end try
                    end repeat
                    if ErrNo is greater than 0 or WordLength = (StartChar - 1) then
                        set MyInput to (item s of Shortcuts)
                        ErrorSafe(MyInput, ErrNo, SubErr)
                    end if
                end if
            end repeat
            -- if CW = "word" then ClickWord()
            set MyInput to OriginalInput
            if NoEntryYet = true then
                set subtask to subtask + 1
                AddTaskNumbersToOutput()
            end if
        end if
    end repeat
end if
end OperativeStep
on JumpToText(MajTask)
    global LookUpOp
    set FindText to MajTask as string
    ClickMenu("TextEdit", "Window", "TaskAnalysis" & LookUpOp, "0")
    -- delay 0.1
    do shell script "sleep " & (0.1) as string
    tell application "System Events"
        keystroke "f" using {command down}
        keystroke (" ".0"
        keystroke return
    end tell
end JumpToText

on LookUpTask()
    global MajTask, MinTask, MyInput, OutputLine, DefinedAlready, Tasks, TasksRef, subtask, LearningMode
    Add0Task(MajTask, MinTask)
    set ParaText to item (MinTask + 1) of item (MajTask) of TasksRef
    -- set CountWords to count words in ParaText
    -- set BuildAction to ""
    -- repeat with i from 1 to CountWords
    -- set BuildAction to BuildAction & (word i in ParaText) & " 
    -- end repeat
    set NewParaText to ParaText as string
    if LearningMode = 1 or (length of NewParaText is less than 6) then
        display dialog "Task number " & MajTask & "." & MinTask & " is " & return & ParaText & return & "Enter new task definition" default answer NewParaText
        set NewParaText to text returned of result
        set item (MinTask + 1) of item (MajTask) of TasksRef to NewParaText
        UpdateTAWindow()
    end if
    set MyInput to NewParaText
    AddTaskNumbersToOutput()
end LookUpTask

on ReadSubTask()
    global MajTask, MinTask, TNRef
    repeat (MajTask - (count of items in TNRef)) times
        copy {} to end of TNRef
    end repeat
    repeat (MinTask + 1 - (count of items in (item MajTask of TNRef))) times
        copy 0 to end of (item MajTask of TNRef)
    end repeat
    set subtask to item (MinTask + 1) of item MajTask of TNRef
    return subtask
end ReadSubTask

on AddTaskNumbersToOutput()
global subtask, OutputLine, MajTask, MinTask, MyInput, CheckMainAction, 
NoEntryYet, TNRef, MinTaskCheck, Outputline2
set NoEntryYet to false
set subtask2 to 0
set MyInput to Abbreviations(MyInput)
if CheckMainAction > 0 then
    set OutputLine to ("" & MajTask & "." & MinTask & ".0:
" & MyInput & ")
else if MinTaskCheck = true then
    set OutputLine to (" " & MajTask & "." & MinTask & ".0:
" & MyInput & ")
else
    set OutputLine to (" " & MajTask & "." & MinTask & "." & subtask & ":
" & MyInput & ")
    set CountTNum to ((MajTask * 10) + MinTask - 9)
    set item (MinTask + 1) of item MajTask of TNRef to subtask
    set subtask2 to subtask
end if
set Outputline2 to OutputLine
if Outputline2 contains "" then set Outputline2 to
ReplaceChars(Outputline2, ",", ",")
AddToData()
end AddTaskNumbersToOutput

on ErrorSafe(CW, ErrNo, SubErr)
global MyInput, PanelQuestion, ErrorPrompt, subtask, OutputLine, 
MajTask, MinTask, StringLength, AllErrors, InstList, InstQuestion, 
WordLength, Shortcuts, ErrorList, Question, ErrorPanel, Inst, InstLabel
set SubErrShort to {"a", "b", "c", "d", "e"}
set SubErrList to {" (fascial)", " (into fat)", " (serosa visible)", " (into rectal
muscle)", " (perforation)"}
set ExpErr to ""
if SubErr is not equal to "" and SubErrShort contains SubErr then
    repeat with s from 1 to (count of items in SubErrShort)
        try
            if (item s of SubErrShort) = SubErr then
                set ExpErr to (item s of SubErrList)
                exit repeat
            end if
        end try
    end repeat
end if
if Shortcuts contains CW then
    repeat with j from 1 to (count of items in Shortcuts)
        if CW = (item j of Shortcuts) then
            -- display dialog "Word is " & CW & return & "ErrNo is " & 
ErrNo
            set ErrorPrompt to ("Type of " & (item j of ErrorList) & ": ")
            set PanelQuestion to ErrorPrompt & return & (item j of Question)
            if ErrNo = 0 then set ErrNo to GetNumber(""")
try
    set ReadErr to (item ErrNo of (item j of AllErrors))
    --display dialog "ReadErr is " & ReadErr
    set ErrNo to ("" & ErrNo & SubErr & ". " & ReadErr & ExpErr)
    on error
        display dialog "Problem reading item " & ErrNo & " of item " & j
    end try
end if
end repeat
else if CW = "inst" then
    set InstPriority to "Primary"
    set Action2 to ""
    repeat with i from 1 to 2
        set ErrorPrompt to "Instrument change" & return & "Select instrument " & i & ": " & return
        set PanelQuestion to ErrorPrompt & return & InstQuesti
        on
            set ErrNo to GetNumber(item i of Inst)
            try
                set (item i of Inst) to ErrNo
                InstWindow(i)
                set Action2 to Action2 & (InstPriority & " = " & ErrNo & ". " & InstLabel)
                if i = 1 then set Action2 to Action2 & "; "
            end try
            set InstPriority to "Secondary"
        end repeat
        set ErrorPrompt to "Instrument change: "
        set ErrNo to Action2
    else
        set ErrNo to "Other: "
    end if
    set AdditionalDetail to ""
    if MyInput is not equal to "inst" then
        if class of ErrNo = integer then set ErrNo to (ErrNo & ". " [Blank entry: please complete via Preferences])
        display dialog "" & ErrorPrompt & ErrNo & return & ". " Enter additional detail:" default answer """ set AdditionalDetail to text returned of result
        end if
        set MyInput to ErrorPrompt & ErrNo
        if AdditionalDetail is not equal to "" then set MyInput to MyInput & AdditionalDetail
    end if
    set subtask to subtask + 1
    AddTaskNumbersToOutput()
end ErrorSafe

on InstWindow(i)
    global Inst, InstList, InstLabel
    set InstLabel to (item (item i of Inst) of InstList)
tell window "Instruments" to set contents of text field (item (i + 2) of Inst) to InstLabel
end InstWindow

on GetNumber(DefAns)
global PanelQuestion, ErrorPrompt, WordLength, MyInput, ErrorPanel
DisplayPanel("ErrorWindow", "Yes", PanelQuestion)
set ErrNo to ""
repeat until class of ErrNo = integer
    display dialog "Select " & ErrorPrompt default answer DefAns
    set ErrNo to (text returned of result)
    try
        set ErrNo to ErrNo as integer
    on error
        display dialog "Enter a number as per Control Panel display"
    end try
end repeat
ClosePanel("ErrorWindow")
return ErrNo
end GetNumber

on AddToData()
global OutputLine, OpRef, PlayPoint, FileText, HighTime, RewoundTime, NewAnalysis
set OutputLineTime to (OutputLine & " " & (round (PlayPoint)) & return)
if NewAnalysis = "Read only" then
    display dialog "Cannot add to completed file (Read only)" with icon "MyScalpel" giving up after 2
else
    write (OutputLineTime) to OpRef starting at eof
end if
set FileText to FileText & OutputLineTime
if PlayPoint is greater than HighTime then set HighTime to PlayPoint
set RewoundTime to PlayPoint
end AddToData

on EditLast()
global OpRef, CurrentFile, OldDelims, ThisFile, ThisFileRef, NewAnalysis
if NewAnalysis = "Read only" then
    display dialog "Cannot edit completed file (Read only)" with icon "MyScalpel" giving up after 2
else
    PauseRoutine()
    try
        close access OpRef
    end try
    set OpRef to (open for access file CurrentFile with write permission)
    set FileAsString to (read OpRef as string)
    set FileAsList to (every paragraph of FileAsString)
    set ThisFile to {}
set ThisFileRef to a reference to ThisFile
repeat with i from 1 to (count of items in FileAsList)
    if (item i of FileAsList) is not equal to "" then copy (item i of FileAsList)
to end of ThisFileRef
end repeat
set TotalParNo to (count of items in ThisFile)
set ParNo to TotalParNo
set EditButtonList to {"OK", "Prev"}
set EditLastButton to "Prev"
repeat
    set ThisPar to item ParNo of ThisFileRef
    if ThisPar contains "TimeElapsed" or ThisPar contains "Date & Time of analysis" or ThisPar = "" then
        else
            set EditLastDial to "Entry number " & ParNo & " is:" & return & return & ThisPar
        DisplayPanel("ErrorWindow", "Yes", EditLastDial)
display dialog "Edit entry " & ParNo & ": " default answer ThisPar buttons EditButtonList default button "OK"
        set EditReply to result
        set EditLastButton to button returned of EditReply
        set NewLine to text returned of EditReply
        ClosePanel("ErrorWindow")
        if EditLastButton = "OK" then exit repeat
        end if
        if EditLastButton = "Prev" and ParNo is greater than 1 then set ParNo to ParNo - 1
        if EditLastButton = "Next" and ParNo is less than TotalParNo then set ParNo to ParNo + 1
        set EditButtonList to {"OK", "Next", "Prev"}
        if ParNo = 1 then
            display dialog "This is the first paragraph in the text" buttons {"OK"}
default button "OK" giving up after 2
            set EditButtonList to {"OK", "Next"}
            set EditLastButton to "Next"
        else if ParNo = TotalParNo then
            display dialog "This is the last paragraph in the text" buttons {"OK"}
default button "OK" giving up after 2
            set EditButtonList to {"OK", "Prev"}
            set EditLastButton to "Prev"
        end if
    end if
end repeat
set (item ParNo) of ThisFileRef to NewLine
set AppleScript's text item delimiters to return
set NewFile to ThisFileRef as string
write NewFile & return to OpRef starting at 0
--close access OpRef
set AppleScript's text item delimiters to OldDelims
end if
end EditLast
on WriteMovieTime()
global TimeElapsed, OpRef, Sync, VLCRate, FileText, OrigText, Sources, RealSource, Cont
if FileText = OrigText then
    set eof of OpRef to 0
    write FileText to OpRef starting at 0
else
    set WriteSync to ""
    repeat with a from 1 to count of items in Sync
        repeat with b from 2 to (count of items in (item a of Sync))
            try
                set ThisSync to (item 1 of (item a of Sync))
                if ThisSync = "Quicktime" then set ThisSync to "QT"
                set ThisSource to ""
                try
                    repeat with c from 1 to 4
                        set CheckSource to (item 1 of item c of Sources)
                        if (item b of item a of RealSource) contains CheckSource then
                            set ThisSource to " (Source = " & CheckSource & ")"
                            exit repeat
                        end if
                    end repeat
                end try
                set WriteSync to WriteSync & " ; Sync" & ThisSync & ThisSource & " = " & (((item b of (item a of Sync)) * 100) as integer) / 100) as text
            end try
        end repeat
    end repeat
    write ("TimeElapsed of completed tracks = " & TimeElapsed & WriteSync & return) to OpRef starting at eof
end if
end WriteMovieTime

on Abbreviations(MyInput)
global OutputLine, Outputline2, DictList, OldDelims
set DictCount to (count of items in DictList) div 2
repeat with i from 1 to DictCount
    set ThisAbbrev to (item ((i * 2) - 1) of DictList)
    set ThisFull to (item (i * 2) of DictList)
    if MyInput contains ThisAbbrev then set MyInput to ReplaceChars(MyInput, ThisAbbrev, ThisFull)
end repeat
set MyInput to CapSent(MyInput)
set OutputLine to MyInput
end Abbreviations
on DisplayPanel(PanelName, AttachState, MyText)
    set ErrorPanel to window PanelName
    if MyText is not equal to "" then
        set NumRows to count of paragraphs in MyText
        set MaxLen to 35
        set LongRows to 0
        repeat with i from 1 to NumRows
            set ParaLen to length of paragraph i of MyText
            if ParaLen is greater than MaxLen then set MaxLen to ParaLen
            set LongRows to LongRows + (ParaLen div 60)
        end repeat
        -- display dialog "This is file " & (paragraph 1 of MyText) & return & "This is line " & i & ", and longest so far is " & MaxLen & return & "Longrows = " & LongRows
        if MaxLen is greater than 60 then set MaxLen to 60
        set WinWide to (MaxLen * 7) + 60
        set NumRows to NumRows + LongRows
        set size of window "ErrorWindow" to {WinWide, (NumRows * 17) + 85}
        tell ErrorPanel
            set contents of text field "ErrorList" to MyText
            set size of text field "ErrorList" to {WinWide - 20, ((NumRows * 17) + 45)}
        end tell
    end if
    if AttachState = "Yes" then
        display pane ErrorPanel attached to window "Control Panel"
    else
        display panel ErrorPanel
    end if
end DisplayPanel

on ClosePanel(PanelName)
    global ErrorPanel
    set ErrorPanel to window PanelName
    close panel ErrorPanel
end ClosePanel

on TextFromList(StartString, myList, ToNum, AddNum, NumCol, StartCol, Delim)
    global OldDelims
    set MyString to StartString
    if MyString is not equal to "" then set MyString to MyString & return
    repeat with i from 1 to (count of items in myList) div NumCol
        if ToNum = "Yes" then set MyString to MyString & (i + AddNum) & "." & Delim
        repeat with j from StartCol to NumCol
            OldDelims = OldDelims & Delim
            set MyString to MyString & (item i * NumCol + j) & OldDelims
        end repeat
    end repeat
end TextFromList
set MyString to MyString & (item (((i - 1) * NumCol) + j) of MyList)
if j is not equal to NumCol then set MyString to MyString & Delim
end repeat
set MyString to MyString & return
end repeat
return MyString
set AppleScript's text item delimiters to OldDelims
end TextFromList

on ListFromText(MyText, StartRow, StartCol, ToNum, Delim, MinLen)
global OldDelims
set MaxCol to 1
set BlankRow to 0
set MyList to {}
set AppleScript's text item delimiters to Delim
repeat with i from StartRow to (count of paragraphs in MyText)
    set ThisPara to paragraph i of MyText
    set numItems to count of text items in ThisPara
    if numItems is greater than MaxCol then set MaxCol to numItems
end repeat
repeat with i from StartRow to (count of paragraphs in MyText)
    set ThisPara to paragraph i of MyText
    if length of ThisPara is less than MinLen then
        set BlankRow to BlankRow + 1
    else
        if ToNum = "Yes" then copy ((i - BlankRow) as text) to end of MyList
        -- if ThisPara is not equal to "" then
        repeat with j from StartCol to MaxCol
            try
                set ThisItem to (text item j of ThisPara)
            on error
                set ThisItem to ""
            end try
            copy ThisItem to end of MyList
        end repeat
    end if
end repeat
return MyList
set AppleScript's text item delimiters to OldDelims
end ListFromText

on EditAbbreviations()
    PauseRoutine()
global OutputLine, OutputLine2, ErrorPath, ErrorList, AbbrevList, DictRef, DictText, OldDelims, DictList
set DictText to TextFromList("Dictionary", DictList, "Yes", 0, 2, 1, "")
set CountEntries to (count of paragraphs in DictText) - 1
DisplayPanel("ErrorWindow", "Yes", DictText)
set AppleScript's text item delimiters to tab
display dialog "Which abbreviation do you wish to edit?" buttons {" Cancel ", " OK " } default button 2 default answer 1
set EditReturn to result
set EditPref to text returned of EditReturn
set EditButton to button returned of EditReturn
if EditButton = " OK " then
    try
        set EditPref to EditPref as integer
    on error
        set EditPref to CountEntries
    end try
    if EditPref is less than 1 or EditPref is greater than CountEntries then set EditPref to CountEntries
    set DictLine to paragraph (EditPref + 1) of DictText
    if (count of text items in DictLine) is greater than 2 then
        set DictAbbrev to text item 2 of DictLine
        set DictFull to text item 3 of DictLine
    else
        set DictAbbrev to ""
        set DictFull to ""
    end if
    set AppleScript's text item delimiters to OldDelims
    display dialog "Abbreviation number " & EditPref & " is (" & DictAbbrev & 
    " )" default answer DictAbbrev
    set DictAbbrev to text returned of result
    display dialog "Expanded text number " & EditPref & " is (" & DictFull & 
    " )" default answer DictFull
    set DictFull to text returned of result
    set NewLine to ("" & EditPref & tab & DictAbbrev & tab & DictFull) as text
    if EditPref = CountEntries then set DictText to DictText & NewLine & 
    return
    set DictText to OverWrite(DictLine, NewLine, DictText, DictRef, 
    ErrorPath)
    set DictList to ListFromText(DictText, 2, 2, "No", ",", 5)
    end if
    ClosePanel("ErrorWindow")
end EditAbbreviations

on ReplaceChars(MyInput, SearchString,ReplacementString)
global OldDelims
if SearchString is not equal to "" then
    set AppleScript's text item delimiters to the SearchString
    set the ItemList to every text item of MyInput
    set AppleScript's text item delimiters to the ReplacementString
    set MyInput to the ItemList as string
    set AppleScript's text item delimiters to OldDelims
end if
return MyInput
end ReplaceChars
on CapSent(MyInput)
    global OldDelims, NewText
    set LowCase to "abcdefghijklmnopqrstuvwxyz"
    set UpCase to "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    set SentenceDelim to {" .", "? " , "! "}
    set CountSD to count of items in SentenceDelim
    repeat with j from 1 to CountSD
        set AppleScript's text item delimiters to (item j of SentenceDelim)
        set AllSent to every text item in MyInput
        set SentCount to count of items in AllSent
        set AppleScript's text item delimiters to OldDelims
        set NewText to {}
        repeat with i from 1 to SentCount
            set ThisChar to character 1 of item i of AllSent
            set OffsetChar to offset of ThisChar in LowCase
            try
                set FirstCap to (character OffsetChar of UpCase)
            on error
                set FirstCap to ThisChar
            end try
            try
                set RestOfSent to (characters 2 thru -1 of (item i of AllSent))
            on error
                set RestOfSent to ""
            end try
            set NewSent to FirstCap & RestOfSent
            copy NewSent to end of NewText
        end repeat
        set AppleScript's text item delimiters to (item j of SentenceDelim)
        set MyInput to NewText as string
    end repeat
    set AppleScript's text item delimiters to OldDelims
    return MyInput
end CapSent

on UpdateTAWindow()
    global filename, FilePath, Tasks, TasksRef, NumParas, TAName,
    CountTA, OpRef, CurrentFile, ArrayPoint, OpNumber
    set TAText to ""
    repeat with m from 1 to (count of TasksRef)
        repeat with n from 0 to (count of items in (item m of TasksRef)) - 1
            set ParaText to (item (n + 1) of item m of TasksRef)
            if n = 0 then
                set TAText to TAText & (" " & m & "." & n & " & ParaText & return)
            else
                set TAText to TAText & (" " & m & "." & n & " & ParaText & return)
            end if
        end repeat
    end repeat
end UpdateTAWindow
end repeat
tell application "TextEdit"
  activate
  set WinList to name of every document
  repeat with i from 1 to (count of items in WinList)
    if (item i of WinList) contains "TaskAnalysis" then set TAWin to i
  end repeat
  try
    set text of document TAWin to TAText
  end try
  save document TAWin
end tell
end UpdateTAWindow

on ReadIOVFile()
  global myPath, OpType, filename, LowNum, HighNum, HighTime,
  IOVRead, IOVStatus, IOVFile, IOVFileRef, IOVTimes, PlayPoint, TimeWin,
  DemoOn, ThisButton, Sync, AddDigit, IOVALready, IOVPara,
  LowestIOVNotDone
  set IOVFile to myPath & "IOV Files:" & filename & " IOV"
  repeat 3 times
    try
      set IOVFileRef to (open for access file IOVFile with write permission)
      exit repeat
    on error
      close access (IOVFile as alias)
      --display dialog "Problem in the error loop" giving up after 1
    end try
  end repeat
  try
    set IOVRead to read file IOVFile
  on error
    set IOVRead to ""
  end try
  if IOVRead = "" then write (filename & " Inter-Observer Validation Report" &
    return & "OpType: " & OpType & return & return) to IOVFileRef starting at 0
  set AppleScript's text item delimiters to "Playback starts at "
  set IOVPlayback to every text item in IOVRead
  set AppleScript's text item delimiters to "Analysis of Episode"
  set IOVEpisode to every text item in IOVRead
  set IOVTimes to {}
  set IOVPara to {}
  set IOVALready to false
  set LowestIOVNotDone to "N/A"
  repeat with i from 1 to (count of items in IOVPlayback) - 1
    try
      set ThisIOV to (item (i + 1) of IOVPlayback)
      set StartTime to (word 1 of ThisIOV) as integer
      set ThisEpisode to (word 1 of (item (i + 1) of IOVEpisode)) as integer
      set EpisodeDiff to ThisEpisode - (count of items in IOVPara)
repeat EpisodeDiff times
    copy 0 to end of IOVTimes
    copy 0 to end of IOVPara
end repeat
repeat with j from 2 to (count of paragraphs in ThisIOV)
    try
        set IsInteger to word 1 of (paragraph j of ThisIOV) as integer
        set (item ThisEpisode of IOVPara) to (item ThisEpisode of
            IOVPara) + 1
    end try
end repeat
if IOVTimes does not contain StartTime then set (item ThisEpisode of
    IOVTimes) to StartTime
end try
end repeat
repeat with i from 1 to (count of items in IOVPara)
    if item i of IOVPara = 0 then
        set LowestIOVNotDone to i
        exit repeat
    else
        set IOVALready to true
    end if
end repeat
set AppleScript's text item delimiters to ""
end ReadIOVFile

on IOV()
    global myPath, OpType, filename, LowNum, HighNum, HighTime,
        IOVRead, IOVStatus, IOVFile, IOVFileRef, IOVTimes, PlayPoint, TimeWin,
        DemoOn, ThisButton, Sync, AddDigit, IOVALready, IOVPara,
        LowestIOVNotDone
    -- Format of IOV output is as follows:
    -- Line 1 = OpName Inter-Observer Validation Report
    -- Line 2 = OpType: Open Anterior resection
    -- Episodes identified as Analysis of Episode 01 on [Date]
    -- Each line = Timepoint TaskNo ARMN Prose
    set IOVHigh to HighTime
    if ThisButton = "IOV2" then set IOVHigh to HighNum
    set lovLow to (((item 2 of (item 1 of Sync)) + (item 2 of (item 2 of Sync))) ^
        2)^ 0.5) + 1
    -- display dialog "Upper limit = " & IOVHigh
    set IOVStatus to "IOV_On"
    tell tab view item "ValidTab" of tab view "TabWin" of window "Control Panel"
        set visible of button "NewObs" to true
        set visible of button "NextEpisode" to true
        set visible of button "ForcePlay" to true
    end tell
    set TimeWin to 300 -- 5 minute episodes
end IOV()
ReadIOVFile()
if IOVALready = true then
  --set IOVPara2 to {}
  --repeat with a from 1 to (count of items in IOVTimes)
  -- if item a of IOVTimes is not equal to 0 then copy (item a of
  IOVPara) to end of IOVPara2
  --end repeat
  set IOVQuestion to TextFromList("Do you wish to re-validate an
episode?" & return, IOVPara, "Yes", 0, 1, 1, " Number of lines = ")
  DisplayPanel("ErrorWindow", "Yes", IOVQuestion)
  set ReDoButton to "OK"
  if LowestIOVNotDone = "N/A" then set ReDoButton to "No"
  display dialog "Re-analyse an episode?" default answer
LowestIOVNotDone buttons {"OK", "No"} default button ReDoButton
  set Entry to result
  ClosePanel("ErrorWindow")
  set ReDoIOV to button returned of Entry
  if ReDoIOV = "OK" then
    try
      set ReDoIOV to (text returned of Entry) as integer
      StartIOV(ReDoIOV)
    on error
      display dialog "Unable to re-analyse episode " & (text returned of
      Entry)
    end try
  end if
else
  exit repeat
end if
end repeat

repeat 40 times
  if (count of items in IOVTimes) is greater than 9 then exit repeat
  set newtime to (random number from IOVLow to IOVHigh) as integer
  if newtime is less than HighTime then
    set UseNewTime to true
    repeat with a from 1 to (count of items in IOVTimes)
      set ThisIOV to item a of IOVTimes
      if (newtime + TimeWin) is greater than ThisIOV and (newtime -
      TimeWin) is less than ThisIOV then
        set UseNewTime to false
        exit repeat
      end if
    end repeat
    if UseNewTime = true then copy newtime to end of IOVTimes
  end if
end repeat
set IOVTimes to SortList(IOVTimes)
repeat with i from 1 to (count of items in IOVTimes)
    if i is greater than (count of items in IOVPara) then copy 0 to end of IOVPara
    if (item i of IOVPara) = 0 then
        StartIOV(i)
        display dialog "Review Next random sequence, or Exit?" buttons {"Next", "Exit"} default button "Next"
        set IOVButton to button returned of result
        if IOVButton = "Exit" then exit repeat
    end if
end repeat
set DemoOn to false
DemoMode(0)
set IOVStatus to "IOV_Off"
tell tab view item "ValidTab" of tab view "TabWin" of window "Control Panel"
    set visible of button "NewObs" to false
    set visible of button "NextEpisode" to false
    set visible of button "ForcePlay" to false
    set contents of text field "StepBox" to ""
end tell
close access (IOVFile as alias)
end IOV

on StartIOV(DoIOV)
    global IOVTimes, newtime, s_newtime, IOVEpisode, IOVFileRef, PlayPoint, AutoSync, s_PlayPoint, ThisButton, IOVStatus, DemoOn, AddDigit, IOVWaiting
    set IOVWaiting to DisplayIOV(false, "None")
    set newtime to (item DoIOV of IOVTimes)
    set s_newtime to CTS(newtime)
    set AddDigit to "0"
    if DoIOV is greater than 9 then set AddDigit to ""
    set IOVEpisode to "Analysis of Episode " & AddDigit & DoIOV
    set DoEpisode to true
    set IOVOutput to "Validated Comment" & newtime & return & IOVEpisode & " on " & (current date) & return & "Playback starts at " & newtime & return & "Time Task Error type Validated Comment" & return
    write IOVOutput to IOVFileRef starting at eof
    -- GoToNewTime using NewTime
    set PlayPoint to newtime
    GoToNewTime()
    -- SyncRoutine(2)
    set s_PlayPoint to CTS(PlayPoint)
    set ThisButton to "IOV"
    set IOVStatus to "IOV_On"
    set DemoOn to true
    DemoMode(newtime)
write ("End of Episode " & (AddDigit & DoIOV) & return) to IOVFileRef
starting at eof
end StartIOV

on SortList(MyList)
    set NewList to {}
    repeat with a from 1 to count of items in MyList
        set NewLow to ""
        repeat with b from 1 to count of items in MyList
            if NewList does not contain (item b of MyList) then
                if NewLow = "" or (item b of MyList) is less than NewLow then set
                    NewLow to (item b of MyList)
                end if
            end if
        end repeat
        copy NewLow to end of NewList
    end repeat
    return NewList
end SortList

on ForceIOV()
    global IOVWaiting
    PauseRoutine()
    repeat 10 times
        if IOVWaiting = "None" then exit repeat
        set T1 to current date
        repeat
            if (current date) is greater than (T1 + 5) then exit repeat
        end repeat
    end repeat
    CheckToPlay()
end ForceIOV

on IOVEntry(IOVButton)
    global IOVFileRef, Outputline, OldDelims, PlayPoint, IOVWaiting, DemoOn
    if IOVButton is not equal to "NewObs" then set IOVWaiting to
    DisplayIOV(false, "None")
    if IOVButton = "NextEpisode" then
        set DemoOn to false
    else
        set IOVProse to ""
        set IOVCATEGORY to "[Routine]"
        set AppleScript's text item delimiters to ": Type of "
        if (count of text items in Outputline) is greater than 1 then set
            IOVCATEGORY to (word 1 of (text item 2 of Outputline)) & space & (word 2 of (word 2 of Outputline))
        set AppleScript's text item delimiters to OldDelims
        if IOVButton = "NewObs" then
            display dialog "Enter new observation for Inter-Observer Validation" buttons {"OK", "Cancel"} default button "OK" default answer ""
set Entry to result
set IOVProse to text returned of Entry
set IOVButton to button returned of Entry
if IOVButton = "OK" then set IOVButton to "Additional comment"
display dialog "Enter Error Classification for this observation (e.g. 'EEM 2'):" buttons {"OK", "None"}) default button "OK" default answer ""
set Entry to result
set IOVCategory to text returned of Entry
if IOVCategory = "" or button returned of Entry = "None" then set IOVCategory to "None"
else if IOVButton = "No comment" or IOVButton = "Accept" then
else
display dialog "Enter observation for Inter-Observer Validation (" & IOVButton & " entry)" buttons {"OK"} default button "OK" default answer ""
set IOVProse to text returned of result
end if
set IOVLine to ("" & (PlayPoint as integer) & tab & (word 1 of Outline2) & tab & IOVCategory & tab & IOVButton & tab & IOVProse) as string
if IOVButton is not equal to "Cancel" then write IOVLine & return to IOVFileRef starting at eof
end if
CheckToPlay()
end IOVEntry

on ShowEEM()
try
ClickMenu("Preview", "Window", "ErrorModes (1 Page)", "0")
ClickMenu("Preview", "Window", "ErrorModes.pdf (1 Page)", "0")
do shell script "sleep " & (6) as string
tell application "System Events" to tell process "Preview" to keystroke "m" using {command down}
ScriptToFront()
end try
end ShowEEM

on ViewText()
global FileText
tell window "TextWin"
set contents of text view "TS" of scroll view "TS" to FileText
set Vis to visible
if Vis = true then set visible to false
if Vis = false then set visible to true
end tell
end

on clicked theObject
    global MyInput, ThisButton, TimeElapsed, LearningMode, TimeSource, Sync, contentSize, moviecount, Cont, SeeQT, MyState, VLCList, PrefsSource, RealSource, Sources, QTRate, filename
set ThisButton to name of theObject
set IOVButtonList to {"Accept", "Modify", "Reject", "NewObs", "NextEpisode"}
tell tab view item "AnalysisTab" of tab view "TabWin" of window "Control Panel"
    set MyInput to contents of text field "CurrEntry"
    set contents of text field "CurrEntry" to ""
end tell
if ThisButton = "ShowEEM" or MyInput = "eem" then
    ShowEEM()
else if MyInput = "qt" or ThisButton = "QTRate" then
    DoQTRate("None", 1)
else if ThisButton contains "Snap" or MyInput = "snap" then
    DoSnap(ThisButton)
else if ThisButton = "SwapSides" or MyInput = "swap" or MyInput = "sp" then
    SwapSides("Clicked")
else if ThisButton = "ViewText" or MyInput = "vt" or MyInput = "text" then
    ViewText()
else if ThisButton = "DoPrefs" or MyInput = "prefs" then
    DoPrefs()
else if ThisButton = "HelpButton" or MyInput = "?" or MyInput = "help" then
    HelpRoutine()
else if ControlButtons contains ThisButton then
    GetMovieStats("Clicked")
    CollectData()
    WriteTextBox()
else if IOVButtonList contains ThisButton then
    IOVEntry(ThisButton)
else if ThisButton = "TextSize" then
    -- ChangeTextSize(1)
else if ThisButton = "EditError" then
    EditErrors()
else if ThisButton = "Abbrev" then
    EditAbbreviations()
else if ThisButton = "EditLast" then
    EditLast()
else if ThisButton = "ClosePanel" then
    ClosePanel("ErrorWindow")
else if ThisButton = "IOV1" or ThisButton = "IOV2" then
    IOV()
else if ThisButton = "ControlVLC" then
    CountVLC()
else if ThisButton = "SyncTracks" then
SyncTracks()
end if
if PrefsButtons contains ThisButton then
  --Only change preferences if one of the PrefsButtons is clicked
  set OldPrefsSource to PrefsSource
  set OldSeeQT to SeeQT
  set AskCont to {"", "", "ControlDV Tape", "ControlVLC", "ControlVHS"}
tell tab view item "PrefsTab" of tab view "TabWin" of window "Control Panel"
    set (item 1 of Cont) to state of button "ControlQT"
    set (item 2 of Cont) to state of button "ControlDVD"
    set (item 3 of Cont) to state of button "ControlDV Tape"
    set (item 4 of Cont) to state of button "ControlVLC"
    set (item 5 of Cont) to state of button "ControlVHS"
    set LearningMode to state of button "Learning"
    set SeeQT to state of button "SeeQT"
    set PrefsSource to current row of matrix "TimeSourcePrefs"
    set PrefsTime to contents of text field "PrefsElapsed"
    set InstView to state of button "Inst"
    set TrackView to state of button "Vid"
  end tell
  if (item 1 of Cont) = 0 and (item 2 of Cont) = 0 then
    AlertSource(ThisButton)
    if (item 1 of Cont) = 1 then CheckForQT()
    if AskCont contains ThisButton then
      repeat with k from 3 to 5
        if (item k of AskCont) = ThisButton and (item k of Cont = 1) then
          set ThisProg to (item 1 of (item k of Sync))
          set ThisSource to ""
          display dialog "Please enter the video displayed on this source (e.g. Head1) (" & ThisProg & ")" default answer ""
          set ThisSource to text returned of result
          if (count of (item k of RealSource)) = 1 then copy "" to end of (item k of RealSource)
          set (item 2 of (item k of RealSource)) to ThisSource
          set TempSource to StripSync(ThisSource)
          set TempSource to item 1 of TempSource
          set TempSync to item -1 of (item k of Sync)
          repeat with c from 1 to 4
            if (item 1 of item c of Sources) = TempSource then
              copy TempSync to end of (item c of Sources)
            exit repeat
          end repeat
        end if
      end repeat
    end if
  end if
  try
    set TimeElapsed to PrefsTime as integer
on error
    set PrefsTime to PrefsTime as text
    set TimeElapsed to CST(PrefsTime)
end try
if ThisButton = "ControlDVTape" and (item 3 of Cont) = 1 then
    try
        DismissBTV()
    end try
    tell application "BTV Pro Carbon" to activate
tell application "System Events"
    tell process "BTV Pro Carbon"
        click button 2 of window "Video Input"
    end tell
end tell
end if
if PrefsSource is not equal to OldPrefsSource then
    set TempSync to {}
    repeat with a from 1 to count of items in Sync
        copy item a of Sync to end of TempSync
    end repeat
    set TimeSourceList to ("Quicktime", "DVD")
    set TimeSource to item PrefsSource of TimeSourceList
    set TimeDiff to (item 2 of (item PrefsSource of Sync))
    repeat with a from 1 to count of items in Sync
        repeat with b from 2 to (count of items in (item a of Sync))
            set ThisSync to (item b of (item a of Sync))
            if a is less than 3 then
                set (item b of (item a of Sync)) to ThisSync - TimeDiff
            else if ThisSync is not equal to 0 then
                set (item b of (item a of Sync)) to ThisSync - TimeDiff
            end if
        end repeat
    end repeat
end if
if SeeQT is not equal to OldSeeQT and moviecount is greater than 0 then
    tell application "QuickTime Player"
        if SeeQT = 0 then set controller type of every movie to none
        if SeeQT = 1 then set controller type of every movie to qtvr
    end tell
end if
if InstView = 0 then set visible of window "Instruments" to false
if InstView = 1 then set visible of window "Instruments" to true
if TrackView = 0 then set visible of window "TrackWindow" to false
if TrackView = 1 then set visible of window "TrackWindow" to true
DoTracks()
end if
end DoTracks()
global Sync, Sources, RealSource
-- display dialog "On populate tracks, RealSource = " & (RealSource as
text)
set TrackList to {Sync, RealSource}
set TrackText to "Video Source SyncName" & return
repeat with a from 1 to (count of items in Sync)
    repeat with b from 2 to (count of items in (item a of Sync))
        set AddText to " & (item 1 of (item a of Sync)) & tab & tab
        repeat with c from 1 to 4
        try
            if (item b of item a of RealSource) contains (item 1 of item c of
Sources) then set AddText to AddText & (item 1 of item c of Sources) & tab
        end try
    end repeat
    repeat with d from 1 to count of TrackList
        set AddText to AddText & (item b of (item a of (item d of
TrackList)))
        on error
            set AddText to AddText & tab
        end try
    end repeat
    set TrackText to TrackText & AddText & return
end repeat
end DoTracks

on AlertSource(ThisButton)
    global Cont
    display dialog "Must have at least one of DVD or Quicktime running" with
icon "MyScalpel.tiff" giving up after 2
    tell tab view item "PrefsTab" of tab view "TabWin" of window "Control
Panel"
        set state of button ThisButton to 1
    end tell
    if ThisButton = "ControlQT" then set (item 1 of Cont) to 1
    if ThisButton = "ControlDVD" then set (item 2 of Cont) to 1
end AlertSource

on SyncTracks()
global moviecount, Cont, Sync, TimeSource, PlayPoint, s_PlayPoint,
OldDelims, RealSource, Sources
PauseRoutine()
GetMovieStats("SyncTracks")
set SyncList to {}
set SyncText to " & return & "Available tracks:" & return
repeat with a from 1 to (count of items in Sync)
    repeat with b from 2 to (count of items in (item a of Sync))
        set ThisProg to (item 1 of (item a of Sync))
        set ThisSync to (item b of (item a of Sync))
        set ThisCont to (item a of Cont)
        set ThisPlay to (PlayPoint - ThisSync)
        set s_ThisPlay to CTS(ThisPlay)
        set IsPrim to ""
        if ThisSync = 0 and ThisCont = 1 then set IsPrim to "(Primary source)"
        try
            set ThisSource to (item b of (item a of Sources))
            copy (ThisProg & " " & (b - 1) & tab & ThisSource & tab & s_ThisPlay & tab & IsPrim & tab) to end of SyncList
            on error
                copy (ThisProg & " " & (b - 1) & tab & tab & tab & s_ThisPlay & tab & IsPrim & tab & "(Inactive)"") to end of SyncList
            end try
        end repeat
    end repeat
end repeat
set SyncText to TextFromList(SyncText, SyncList, "Yes", 0, 1, 1, tab)
DisplayPanel("Errorwindow", "Yes", SyncText)
display dialog "Please choose 1st of 2 sources to swap" default answer ""
set Swap1 to (text returned of result) as integer
display dialog "Please choose 2nd of 2 sources to swap" default answer ""
set Swap2 to (text returned of result) as integer
set MySwap to {Swap1, Swap2}
set SwapSync to {0, 0}
set CountSwap to 0
repeat with a from 1 to (count of items in Sync)
    repeat with b from 2 to (count of items in (item a of Sync))
        set CountSwap to CountSwap + 1
        repeat with i from 1 to 2
            if (item i of MySwap) = CountSwap then set (item (3 - i) of SwapSync) to (item b of (item a of Sync))
        end repeat
    end repeat
end repeat
set CountSwap to 0
repeat with a from 1 to (count of items in Sync)
    repeat with b from 2 to (count of items in (item a of Sync))
        set CountSwap to CountSwap + 1
        repeat with i from 1 to 2
            if (item i of MySwap) = CountSwap then set (item b of (item a of Sync)) to (item i of SwapSync)
        end repeat
    end repeat
end repeat
ClosePanel("Errorwindow")
end SyncTracks
on action QTSlider
    global moviecount, Cont
    if name of QTSlider = "QTVol" then
        tell tab view item "PrefsTab" of tab view "TabWin" of window "Control Panel"
            set QTVol to contents of slider "QTVol"
        end tell
        if moviecount is greater than 0 and (item 1 of Cont) = 1 then
            tell application "QuickTime Player"
                set sound volume of movie 1 to (QTVol as integer)
            end tell
        else if (item 2 of Cont) = 1 then
            tell application "DVD Player"
                set audio volume to (QTVol as integer)
            end tell
        end if
    end if
end action

on resized theObject
    DoSize(theObject)
end resized

on will resize theObject
    DoSize(theObject)
end will resize

on DoSize(theObject)
    tell theObject
        set {X1, Y1, X2, Y2} to bounds
        tell scroll view "TS" to set size to {X2 - X1, Y2 - Y1 - 22}
    end tell
end DoSize

on choose menu item theObject
    set N to name of theObject
    if N = "Help" then HelpRoutine()
    if N = "Find" or N = "Next" then FindText(N)
end choose menu item

on end editing theObject
    (*Add your script here.*)
end end editing

on FindText(N)
    global OldDelims, FileText, MyFind, WhichFind
    if N = "Find" then
        set MyFind to text returned of (display dialog "What text to find" default answer "")
end on
set WhichFind to 1
end if
try
  if MyFind is not equal to "" then
    set CountPar to count of paragraphs in FileText
    repeat with i from WhichFind to CountPar
      set ThisPar to paragraph i of FileText
      if ThisPar contains MyFind then exit repeat
    end repeat
    if i is less than CountPar then
      set TOff to offset of ThisPar in FileText
      set TOff2 to (TOff+(offset of MyFind in ThisPar))-2
      tell text view "TS" of scroll view "TS" of window "TextWin"
        call method "scrollRangeToVisible:" of object it with parameter {TOff, TOff}
      end tell
      call method "setSelectedRange:" of object it with parameter {TOff2, TOff2 +(length of MyFind)}
    end tell
  else
    set WhichFind to i+1
  end if
end if
end try
end FindText
on HelpRoutine()
global myPath
Pauseroutine()
try
  tell application "Finder"
    open (myPath & "HelpDocument.pdf") as alias
  end tell
on error
  display dialog "Help Document not found!" giving up after 2
end try
end HelpRoutine
on DoPrefs()
global myPath, OldDelims
set ReadPrefs to read ((myPath & "Preferences") as alias)
set AppleScript's text item delimiters to ""
set PrefList to {}
set PrefSections to (every text item in ReadPrefs)
set PrefCount to (count of items in PrefSections)
set AppleScript's text item delimiters to OldDelims
repeat with i from 3 to (PrefCount - 1)
set ThisPref to (paragraph 1 of item i of PrefSections)
    if length of ThisPref is greater than 50 then set ThisPref to (characters 1 thru 48 of ThisPref) & "...
    copy ThisPref to end of PrefList
end repeat
set PrefText to TextFromList("Choose a preference setting to edit: ",
        PrefList, "Yes", 0, 1, 1, "")
DisplayPanel("ErrorWindow", "Yes", PrefText)
repeat
    display dialog "Choose a preference setting to edit:" default answer 1
    try
        set Entry to text returned of result as integer
        if Entry is greater than 0 and Entry is less than ((count of items in PrefList) + 1) then exit repeat
    on error
        display dialog "Please enter a number between 1 and " & (count of items in PrefList) with icon "MyScalpel" giving up after 2
    end try
end repeat
ClosePanel("ErrorWindow")
set ThisSection to item (Entry + 2) of PrefSections
set PrefData to ListFromText(ThisSection, 1, 1, "No", " ", 3)
    set PrefMinusTitle to (items 2 thru -1 of PrefData)
    if (count of items in PrefData) is greater than 2 then
        set PrefDialog to TextFromList("Please select item from Preference number " & Entry & ": " & return & (item 1 of PrefData) & return,
            PrefMinusTitle, "Yes", 0, 1, 1, "")
        DisplayPanel("ErrorWindow", "Yes", PrefDialog)
        repeat
            display dialog "Please choose an entry to edit (enter " & (count of items in PrefData) & " to add new category)." default answer 1
            try
                set PrefToEdit to (text returned of result as integer)
                if PrefToEdit is greater than 0 then exit repeat
            on error
                display dialog "Please enter a number between 1 and " & (count of items in PrefData) with icon "MyScalpel" giving up after 2
            end try
        end repeat
        ClosePanel("ErrorWindow")
        set DialogPart2 to "Entry number " & PrefToEdit & " is:" 
    else
        set PrefToEdit to 1 
        set DialogPart2 to "Current entry is:" 
    end if
    if PrefToEdit is less than (count of items in PrefData) then
        set OldPref to (item PrefToEdit of PrefMinusTitle)
        set PrefEditDialog to "Preference number " & Entry & " is:" & return & (item 1 of PrefData) & return & return & DialogPart2 & return & OldPref
DisplayPanel("ErrorWindow", "Yes", PrefEditDialog)
display dialog "Please enter new data for Entry " & PrefToEdit & ":"
default answer OldPref
set NewPref to text returned of result
set (item (PrefToEdit + 1) of PrefData) to NewPref
else
set PrefToEdit to count of items in PrefData
set DialogPart2 to "Entry number " & PrefToEdit & " is:" 
set OldPref to (item -1 of PrefMinusTitle)
set PrefEditDialog to "Preference number " & Entry & " is:" & return & (item 1 of PrefData) & return & return & DialogPart2 & return & "(currently blank)"
DisplayPanel("ErrorWindow", "Yes", PrefEditDialog)
display dialog "Please enter new data for additional entry " & PrefToEdit & ":"
default answer ""
set NewPref to text returned of result
copy NewPref to end of PrefData
end if
set NewSection to TextFromList("", PrefData, "No", 0, 1, 1, ")
set ReadPrefs to OverWrite(ThisSection, NewSection, ReadPrefs, "Preferences", myPath)
ClosePanel("ErrorWindow")
end DoPrefs

on EditErrors()
PauseRoutine()
global ErrorPath, ErrorList
set CountErrorList to (count of items in ErrorList)
set EditDialog to TextFromList("Choose a category to edit:", ErrorList, "Yes", 0, 1, 1, ")
DisplayPanel("ErrorWindow", "Yes", EditDialog)
repeat
display dialog "Which category do you wish to edit?" buttons {" Cancel ", " OK "} default button 2 default answer 1
set EditReturn to result
set EditPref to text returned of EditReturn
set EditButton to button returned of EditReturn
try
set EditPref to EditPref as integer
if EditPref is greater than 0 and EditPref is less than (CountErrorList + 1) then exit repeat
on error
display dialog "Please enter a number between 1 and " & CountErrorList with icon "MyScalpel"
end try
end repeat
if EditButton = " OK " then
set EditFile to item EditPref of ErrorList
tell application "TextEdit"
open alias (ErrorPath & EditFile)
set bounds of window 1 to {10, 610, 700, 1024}
edt tell
display dialog "Edit Text File, save and close. Click OK when finished" & return & "*** It is important to retain formatting *** & return & " I.e. Serial numbering and [tab] and 1 " & (item EditPref of ErrorList) & " type per line" & return
tell application "TextEdit"
  close document EditFile saving ask
tell application "QuickTime Player"
  set RightQ to 0
  repeat with i from 1 to count of movies
    set ThisQ to bounds of window 1 of movie i
    if (item 3 of ThisQ) is greater than RightQ then
      set RightQ to (item 3 of ThisQ)
      set Q to i
    else if item 3 of ThisQ = RightQ then
      if name of movie i contains "Head2" or name of movie i contains "Head 2" then set Q to i
  end repeat
  set QList to {3 - Q,Q}
  if (RightQ - DispX)^2 is greater than 400 OR AutoClick = "Clicked" then
    repeat with i from 1 to count of movies
      tell movie (item i of QList)
        if name does not contain "Head1" then set sound volume to 0
        set controller type to none
        set MovX to (item 3 of item i of MovWin) - (item 1 of item i of MovWin)
set dimensions to {MovX, MovX / QRatio}
set bounds of window 1 to item i of MovWin
end tell
end repeat
end if
end tell
-- if QTRight = true then
-- set QTRight to false
-- else
-- set QTRight to true
-- end if
end SwapSides

on SizeWindow(AppName, Win, X1, Y1, X2, Y2)
tell application AppName
    set bounds of window Win to {X1, Y1, X2, Y2}
end tell
end SizeWindow

on ChangeTextSize(z)
global TextWin, moviecount, Cont, QTRight, QTKey, DVDSize, ButtonTitle, DispX, DispY, VLCList
if TextWin = "AllLarge" then
    set QTKey to "1"
    set DVDSize to "Normal"
    set ButtonTitle to "Small Quicktime"
    set TextWin to "QTSmall"
else if TextWin = "QTSmall" then
    set QTKey to "0"
    set DVDSize to "Normal"
    set ButtonTitle to "Small DVD"
    set TextWin to "DVDSmall"
else if TextWin = "DVDSmall" then
    set QTKey to "1"
    set DVDSize to "Half"
    set ButtonTitle to "All Small"
    set TextWin to "AllSmall"
else
    set QTKey to "0"
    set DVDSize to "Half"
    set ButtonTitle to "All Large"
    set TextWin to "AllLarge"
end if
--end if
set XList to {768, 384}
set YList to {598, 310}
if QTKey = "1" then
    set MovX to item 1 of XList
    set MovY to item 1 of YList
else
set MovX to item 2 of XList
set MovY to item 2 of YList
end if
if (item 4 of Cont) = 1 then
set VLCX to MovX
set VLCY to MovY
if moviecount is greater than 0 and (item 1 of Cont) = 1 then
set VLCX to item 1 of XList
set VLCY to item 1 of YList
if DVDSize = "Half" then
set VLCX to item 2 of XList
set VLCY to item 2 of YList
end if
end if
end if
if (count of items in VLCList) is greater than 1 or (moviecount is greater than 0 and (item 2 of Cont) = 1) then
set VLCX to (item 2 of XList)
set VLCY to (item 2 of YList)
end if
end if
set AllWin to {{1, 1, 497, 284}}
if QTRight = true then
if moviecount is greater than 0 and (item 1 of Cont) = 1 and z = 1 then
SizeWindow("QuickTime Player", 1, (DispX - MovX), 0, DispX, MovY)
if (item 4 of Cont) = 1 and z = 1 then
tell application "VLC"
repeat with i from 1 to count of windows
if name of window i does not contain "Controller" then exit repeat
end repeat
tell application "VLC" to set VWin to bounds of window i
end tell
if (item 1 of Cont) = 0 then
SizeWindow("VLC", i, (DispX - VLCX), 0, DispX, VLCY)
else if (item 2 of Cont) = 0 then
SizeWindow("VLC", i, 0, (DispY - VLCY), VLCX, DispY)
else
SizeWindow("VLC", i, (DispX - (item 2 of XList)), (DispY - VLCY), DispX, DispY)
end if
tell application "VLC" to set VWin to bounds of window i
copy VWin to end of AllWin
end if
else
if moviecount is greater than 0 and (item 1 of Cont) = 1 and z = 1 then
SizeWindow("QuickTime Player", 1, 0, (DispY - MovY), MovX, DispY)
if (item 4 of Cont) = 1 and z = 1 then
tell tab view item "PrefsTab" of tab view "TabWin" of window "Control Panel"
set MyState to contents of combo box "DeIntBox"
set visible of text field "DeIntText" to true
set visible of combo box "DeIntBox" to true
end tell
ClickMenu("VLC", "Video", "Deinterlace", MyState)
tell application "VLC"
    repeat with i from 1 to count of windows
        if name of window i does not contain "Controller" then exit repeat
    end repeat
end tell
if (item 1 of Cont) = 0 then
    SizeWindow("VLC", i, 0, (DispY - VLCY), VLCX, DispY)
else if (item 2 of Cont) = 0 then
    SizeWindow("VLC", i, (DispX - VLCX), 0, DispX, VLCY)
else
    SizeWindow("VLC", i, (DispX - (item 2 of XList)), (DispY - VLCY), DispX, DispY)
end if
tell application "QuickTime Player"
    repeat with m from 1 to moviecount
        set QWin to bounds of window 1 of movie m
        copy QWin to end of AllWin
    end repeat
end tell
tell window "Control Panel" to set position to {1, DispY}
set TextW to 0
set TextX to MovX
set TextY to MovY
if (item 2 of Cont) = 1 then
tell application "DVD Player"
    set info visibility to false
    if DVDSIZE = "Normal" and z = 1 then set viewer size to normal
    if DVDSIZE = "Half" and z = 1 then set viewer size to half
    set DVDDWin to viewer bounds
    set DVDX to (item 3 of DVDDWin) - (item 1 of DVDDWin)
    set DVDY to (item 4 of DVDDWin) - (item 2 of DVDDWin)
    if z = 0 then
        else if QTRight = true then
            set TextX to DVDX
            set TextY to MovY
        else
            set TextX to MovX
            set TextY to DVDY + 22
        end if
    if (DispY - TextY) is less than 300 then set TextY to (TextY - 74)
    if QTRight = true and z = 1 then
        set viewer position to {4, (19 + DispY - DVDY)}
    else if z = 1 then
        set viewer position to {((DispX + 10 - DVDX), 22)}
end if
if QTRight = true then
    set ControlY to (DispY - (DVDY + 84))
    set RightControlY to (MovY + 22)
else
    set ControlY to (DispY - (MovY + 84))
    set RightControlY to (DVDY + 22)
end if
if ControlY > 290 then
    set ControlX to 0
else if DVDSize = "Half" or QTKey = "0" then
    set ControlX to (DispX - 348)
    set ControlY to RightControlY
    set TextY to TextY + 102
    -- else
    -- set HorizVert to "Vert"
    -- set ControlX to (DispX - 116)
    -- set ControlY to RightControlY
    -- set TextW to 116
end if
if (item 4 of Cont) = 0 then
    if (item 1 of Cont) = 0 or (item 2 of Cont) = 0 then
        set TextY to 22
    end if
end if
if z = 1 then set controller position to {ControlX, ControlY}
set DWin to viewer bounds
copy DWin to end of AllWin
end tell
end if
set DispX to 1680
set DispY to 1050
set AllWin to {{1, 1, 497, 284}}

tell application "DVD Player"
    set DWin to viewer bounds
    copy DWin to end of AllWin
end tell
tell application "QuickTime Player"
    set m to count of movies
    repeat with i from 1 to m
        set b to bounds of window 1 of movie i
        copy b to end of AllWin
    end repeat
end tell
--return AllWin
set XLow to {{0, 0, 22}}
set XHigh to {{0, 0, DispY}}
repeat with i from 1 to count of AllWin
    set ThisWin to item i of AllWin
set Y1 to item 2 of ThisWin
set Y2 to item 4 of ThisWin
set YMid to Y1 + ((Y2 - Y1) / 2)
set X1 to item 1 of ThisWin
set X2 to item 3 of ThisWin
if X1 is less than 0 then set X1 to 0
if X2 is greater than DispX then set X2 to DispX
if YMid is less than (DispY / 2) then
   --Window is near top
   copy {X1, X2, Y2} to end of XHigh
else
   --Window is near bottom
   copy {X1, X2, Y1} to end of XLow
end if
end repeat
copy {DispX, DispX, 22} to end of XLow
copy {DispX, DispX, DispY} to end of XHigh

set XHigh to XHigh
set XLow to XLow
set XLow to SortWin(XLow)
set XHigh to SortWin(XHigh)
--return XLow
set MaxArea to -1
repeat with a from 1 to (count of XLow) - 1
   set X1 to (item 2 of (item a of XLow))
   set X2 to (item 1 of (item (a + 1) of XLow))
   set OldY to 0
   repeat with b from 1 to (count of XHigh) - 1
      set X3 to (item 1 of (item b of XHigh))
      set X4 to (item 2 of (item b of XHigh))
      set NewY to item 3 of (item b of XHigh)
      -- display dialog "Gap " & a & " is from " & X1 & " to " & X2 & " & return & "Upper movie " & b & " lies between " & X3 & " and " & X4 & ", height " & NewY
      if (X3 is greater than or equal to X1 and X3 is less than X2) or (X4 is greater than X1 and X4 is less than or equal to X2) or (X3 is less than X1 and X4 is greater than X2) then
         if NewY is greater than OldY and NewY is less than DispY then set OldY to NewY
      --display dialog "This is item " & a & " of XLow (" & ((item a of XLow) as text) & ")" & return & "Bounds of window " & b & " are " & NewWin as text
      end if
   end repeat
set NewWin to {X1, OldY, X2, DispY}
set NewArea to ((item 3 of NewWin) - (item 1 of NewWin)) * ((item 4 of NewWin) - (item 2 of NewWin))
if NewArea is greater than MaxArea or NewY is greater than OldY then set OldY to NewY

set MaxArea to NewArea
set XY to NewWin
end if
end repeat
repeat with a from 1 to (count of XHigh) - 1
  set X1 to (item 2 of (item a of XHigh))
  set X2 to (item 1 of (item (a + 1) of XHigh))
  set OldY to DispY
  repeat with b from 1 to (count of XLow) - 1
    set X3 to (item 1 of (item b of XLow))
    set X4 to (item 2 of (item b of XLow))
    set NewY to item 3 of (item b of XLow)
    if (X3 is greater than or equal to X1 and X3 is less than X2) or (X4 is greater than X1 and X4 is less than or equal to X2) or (X3 is less than X1 and X4 is greater than X2) then
      if NewY is less than OldY and NewY is greater than 22 then set OldY to NewY
    end if
  end repeat
  set NewWin to {X1, 22, X2, OldY}
  set NewArea to ((item 3 of NewWin) - (item 1 of NewWin)) * ((item 4 of NewWin) - (item 2 of NewWin))
  if NewArea is greater than MaxArea then
    set MaxArea to NewArea
    set XY to NewWin
  end if
end repeat
SizeWindow("TextEdit", 1, item 1 of XY, item 2 of XY, item 3 of XY, item 4 of XY)

on GetItem(MyList, Num)
  --if Num = 0 then gets all of list
  set NewList to {}
  repeat with i from 1 to count of MyList
    copy (item i of MyList) to end of NewList
  end repeat
  if Num = 0 then
    return NewList
  else
    if Num is greater than i then set Num to i
    return item Num of NewList
  end if
end GetItem
on SortWin(MyList)
    set the index_list to {}
    set the sorted_list to {}
    repeat (count of items of MyList) times
        set the LowNo to 10000
        repeat with i from 1 to (count of items of MyList)
            if i is not in the index_list then
                set ThisItem to item i of MyList
                if (item 1 of ThisItem) is less than LowNo then
                    set the LowNoItem to ThisItem
                    set the LowNo to item 1 of ThisItem
                    set the LowNo_index to i
                end if
            end if
        end repeat
        set the end of sorted_list to the LowNoItem
        set the end of the index_list to the LowNo_index
    end repeat
    return sorted_list
end SortWin

on StoreOldSync()
    global Ti
    meElapsed, Sync, VLCRate, OldSync
    set OldSync to {TimeElapsed}
    copy Sync to end of OldSync
    copy VLCRate to end of OldSync
end StoreOldSync

on RetrieveOldSync()
    global TimeElapsed, Sync, VLCRate, OldSync
    set TimeElapsed to item 1 of OldSync
    set Sync to item 2 of OldSync
    set VLCRate to item 3 of OldSync
end RetrieveOldSync

on ToggleDemo()
    global DemoOn
    if DemoOn = false then
        set DemoOn to true
    else
        set DemoOn to false
    end if
end ToggleDemo

on DemoMode(IOVTime)
    global DemoOn, AutoSync, FileText, PlayPoint, s_PlayPoint, OldDelims,
    MajTask, MinTask, Outputline2, YellowList, IOVStatus, IOVFileRef, TimeWin,
    ThisButton, IOVWaiting, TabName, s_TextTime
tell tab view item "PrefsTab" of tab view "TabWin" of window "Control Panel"
    set SyncDemo to state of button "UseSync"
end tell
set visible of text field "LabelDemo" of tab view item "AnalysisTab" of tab view "TabWin" of window "Control Panel" to true
tell tab view item "ValidTab" of tab view "TabWin" of window "Control Panel"
    set title of button "DemoMode" to "Stop demo"
    set visible of text field "CurrTime" to DemoOn
    set visible of text field "CurrTimeLabel" to DemoOn
    set visible of text field "NextTime" to DemoOn
    set visible of text field "NextTimeLabel" to DemoOn
end tell
if DemoOn = false then set title of button "DemoMode" of tab view item "ValidTab" of tab view "TabWin" of window "Control Panel" to "Demo mode"
if DemoOn = true then
    if SyncDemo = 1 then StoreOldSync()
    SyncRoutine(3)
    CheckToPlay()
    ScriptToFront()
repeat
    set ReStart to false
tell tab view item "ValidTab" of tab view "TabWin" of window "Control Panel" to set contents of text field "CurrTime" to s_PlayPoint
    set OldTime to PlayPoint
repeat with d from 3 to (count of paragraphs in FileText)
    if d = (count of paragraphs in FileText) then exit repeat
    set DemoPara to paragraph d of FileText
    if DemoPara contains "TimeElapsed" then
        if SyncDemo = 1 then ReadSyncLine(DemoPara)
        else if DemoPara does not contain "Date & Time" and DemoPara contains " " then
            try
                set TextTime to last word of DemoPara as integer
            end try
            if TextTime is greater than OldTime then
                repeat with e from d + 1 to (count of paragraphs in FileText)
                    set NextPara to paragraph e of FileText
                    if NextPara does not contain "TimeElapsed" and NextPara does not contain "Date & Time" and NextPara contains " " then exit repeat
                end repeat
                try
                    set NextTime to last word of NextPara as integer
                end try
                set ToForce to (NextTime - TextTime)
                if IOVStatus = "IOV_On" and PlayPoint is greater than (IOVTime + TimeWin) then exit repeat
set s_TextTime to CTS(TextTime)
if TabName = "ValidTab" then set contents of text field "NextTime" of tab view item TabName of tab view "TabWin" of window "Control Panel" to s_TextTime
set DemoColour to "GreenDemo"
if DemoPara contains " Type of " then set DemoColour to "RedDemo"
set DemoColour to "OrangeDemo"
if DemoPara contains "Type of Error: 1." then set DemoColour to "OrangeDemo"
if DemoPara contains "Type of Error: 2." then set DemoColour to "OrangeDemo"
repeat with y from 1 to (count of items in YellowList)
if DemoPara contains ("Type of " & (item y of YellowList) & ": ") then set DemoColour to "YellowDemo"
end repeat
repeat
if DemoOn = false then exit repeat
if (TextTime - PlayPoint) is less than 10 and (TextTime - PlayPoint) is greater than -15 then exit repeat
if PlayPoint is less than OldTime or PlayPoint is greater than (NextTime + 20) then
set ReStart to true
exit repeat
end if
set T1 to current date
repeat
if (current date) is greater than (T1 + 3) then exit repeat
end repeat
end if
if DemoOn = false then exit repeat
if IOVWaiting = "Force response" then ForceIOV()
if IOVWaiting = "No comment" then IOVEntry(IOVWaiting)
if ReStart = true then exit repeat
set word1 to word 1 of DemoPara
set AppleScript's text item delimiters to "."
set MajTask to text item 1 of word1 as integer
set MinTask to text item 2 of word1 as integer
set AppleScript's text item delimiters to OldDelims
set Outputline2 to RemoveTabSpace(DemoPara)
try
ColourBox(DemoColour, "Show")
WriteTextBox()
if IOVStatus = "IOV_On" then
if IOVWaiting = "None" then set IOVWaiting to DisplayIOV(true, "No comment")
if DemoColour = "RedDemo" and ToForce is greater than 0 then set IOVWaiting to "Force response"
end if
set T1 to current date
repeat
if (current date) is greater than (T1 + 3) then exit repeat
   ColourBox(DemoColour, "Hide")
end try
end if
end if
end repeat
if ReStart = false or DemoOn = false then
   set DemoOn to false
   if d = (count of paragraphs in FileText) then display dialog "End of
   Text File" attached to window "Control Panel" with icon "MyScalpel" giving up
   after 2.5
   exit repeat
end if
end if
end repeat
if SyncDemo = 1 then RetrieveOldSync()
end if
HideBoxes(TabName)
set IOVWaiting to DisplayIOV(false, "None")
end DemoMode

on ColourBox(DemoColour, ShowHide)
   global Outline2, TabName, s_TextTime
   try
tell tab view item TabName of tab view "TabWin" of window "Control
   Panel"
      set contents of text field "StepBox" to Outline2 as string
      set contents of text field "CurrTime" to s_TextTime
      if ShowHide = "Show" then
         set visible of text field DemoColour to true
      else
         set MyCol to {"Red", "Yellow", "Orange", "Green"}
         repeat with c from 1 to 4
            set ThisField to "" & (item c of MyCol) & "Demo"
            set visible of text field ThisField to false
         end repeat
      end if
   end tell
end try
end ColourBox

on RemoveTabSpace(ThisLine)
   global OldDelims
   set AppleScript's text item delimiters to tab
   try
      set LineAsList to text items 1 thru -2 of ThisLine
      set ThisTime to (last text item in ThisLine) as integer
      set ThisTime to (CTS(ThisTime))
      set AppleScript's text item delimiters to space
      set ThisLine to (LineAsList as string) & " " & ThisTime
on error
    set ThisLine to ""
end try
set AppleScript's text item delimiters to OldDelims
return ThisLine
end RemoveTabSpace

on ClickWord()
    global filename, s_PlayPoint, ToPaste, PasteSource, FAilClick, ReviewRef, OpRef, Outline2, OldDelims, ReviewCol, PlayPoint, PasteWord, HighTime, EEMName, Sources
    if PasteWord = "Run" then
        GetMovieStats("Word")
        set PasteTime to PlayPoint
        try -- If Word activation is within 15 secs of last entry, will use old time
            if (PlayPoint - HighTime) ^ 2 is less than 225 then set PasteTime to
        end try
        set AppleScript's text item delimiters to ": & tab
        set ToPaste to last text item of Outputline2
        set PasteTask to (word 1 of text item 1 of Outputline2)
        set AppleScript's text item delimiters to ": 
        set ToPaste to last text item of ToPaste
        set AppleScript's text item delimiters to tab
        set ToPaste to first text item of ToPaste
        set AppleScript's text item delimiters to OldDelims
        ScriptToFront() 
        display dialog "Enter comment for Word document:" default answer ToPaste
        set ToPaste to text returned of result
        set PFile to filename as text
        set PasteList to {}
        repeat with n from 1 to count of items in Sources
            copy item 1 of item n of Sources to end of PasteList
        end repeat
        ScriptToFront()
        set PasteSource to choose from list PasteList with prompt "Please choose time source on which this is visible"
        set PlayInt to (CTS(PasteTime as integer))
        set PasteString to ("" & PFile & tab & PasteSource & tab & PasteTask & tab & PlayInt & tab & ToPaste & tab & return) as string
        write PasteString to ReviewRef starting at eof
        else if PasteWord = "End" then
            tell application "TextEdit" to save every document
        end if
    end ClickWord

on WriteTextBox()
    global MajTask, MinTask, Tasks, TasksRef, Outline2, s_PlayPoint, OldMajTask, OldMinTask, OldOut2
if OldMajTask is not equal to MajTask then
    try
        set DialogLine1 to (MajTask & ".0  " & (item 1 of item MajTask of TasksRef))
        on error
            set DialogLine1 to (MajTask & ".0  " & "is not on the current Task Analysis")
        end try
    tell tab view item "AnalysisTab" of tab view "TabWin" of window "Control Panel" to set contents of text field "MajBox" to DialogLine1 as string
end if

if OldMinTask is not equal to MinTask then
    try
        set DialogLine2 to (MajTask & ".  " & MinTask & "  " & (item (MinTask + 1) of item MajTask of TasksRef))
        on error
            set DialogLine2 to (MajTask & ".  " & MinTask & "  " & "is not on the current Task Analysis")
        end try
    tell tab view item "AnalysisTab" of tab view "TabWin" of window "Control Panel" to set contents of text field "SubBox" to DialogLine2 as string
end if

if OldOut2 is not equal to Outputline2 then set contents of text field "StepBox" to Outputline2 as string

set contents of text field "CurrTime" to s_PlayPoint
end tell

set OldMajTask to MajTask
set OldMinTask to MinTask
set OldOut2 to Outputline2
end WriteTextBox

on OverWrite(OldString, NewString, MyFileText, MyFileName, MyFilePath)
    if OldString is not equal to NewString then
        set MyFileText to ReplaceChars(MyFileText, OldString, NewString)
        set PathAndFile to (MyFilePath & MyFileName)
        set FileRef to (open for access alias PathAndFile with write permission)
        set eof of FileRef to 0
        write MyFileText to FileRef starting at 0
        close access alias PathAndFile
    end if
    return MyFileText
end OverWrite

on ClickMenu(app_name, menu_name, menu_item, submenu_item)
    try
        -- bring the target application to the front
        tell application app_name to activate
    on error

tell application "System Events" to tell process app_name to set
  frontmost to true
end try
try
  tell application "System Events" to tell process app_name to tell menu
  bar 1 to tell menu bar item menu_name to tell menu menu_name
    if submenu_item = "0" then
      click menu item menu_item
    else
      tell menu item menu_item to tell menu menu_item
        click menu item submenu_item
      end tell
    end if
end tell
end try
end ClickMenu

on ExtractFromList(MyList)
  set CtList to (count of items in MyList)
  set CtComb to 1
  repeat with i from 1 to CtList
    set CtComb to (CtComb * (count of items in (item i of MyList)))
  end repeat
  set CompiledList to {}
  repeat with a from 0 to (CtComb - 1)
    set MyStr to "" 
    repeat with i from 1 to CtList
      set ThisDiv to 1
      set ThisMod to 1
      repeat with j from (i + 1) to CtList
        set ThisDiv to (ThisDiv * (count of items in (item j of MyList)))
      end repeat
      repeat with k from i to CtList
        set ThisMod to (ThisMod * (count of items in (item k of MyList)))
      end repeat
      set ThisItem to (a mod ThisMod)
      set ItemNo to (ThisItem div ThisDiv)
      set MyStr to MyStr & (item (ItemNo + 1) of (item i of MyList))
      if i is not equal to CtList then set MyStr to MyStr & " "
    end repeat
    copy MyStr to end of CompiledList
  end repeat
  return CompiledList
end ExtractFromList

on DoSnap(ThisButton)
  global PicPath, OldDelims, filename, MajTask, MinTask, PicNum, PlayPoint
  set AppleScript's text item delimiters to "/"
if ThisButton = "SnapPath" or PicPath = "" then set PicPath to (choose folder with prompt "Choose a destination for pictures")
if ThisButton = "TakeSnap" or ThisButton = "Playbutton" then
  set PicName to "" & filename & " " & MajTask & "." & MinTask & " " & (PlayPoint as integer)
  display dialog "Please enter name for picture: " & PicPath & " " default answer PicName
  set PicName to text returned of result
tell application "Finder"
    set AppleScript's text item delimiters to ":"
    set CheckPic to (every file in (PicPath as alias)) as text
    set AppleScript's text item delimiters to ""
    set PicNum to 0
    set PicSp to ""
    repeat
      set PicFull to (PicPath & PicName & PicSp & ".jpg") as text
      if CheckPic does not contain PicFull then exit repeat
      set PicNum to PicNum + 1
      set PicSp to " " & PicNum
    end repeat
    end tell
    do shell script ("screencapture -i " & quoted form of POSIX path of PicFull)
    end if
  set AppleScript's text item delimiters to OldDelims
end DoSnap

on CloseFile()
global myPath, filename, FilePath, Tasks, TasksRef, NumParas, TAName,
CountTA, OpRef, CurrentFile, ReviewFile, ReviewRef, ArrayPoint,
OpNumber, PasteWord, NewAnalysis, OldPic, NewPic, DispX, DispY
  set PasteWord to "End"
  ClickWord()
  if NewAnalysis is not equal to "Read only" then WriteMovieTime()
  UpdateTAWindow()
  close access OpRef
  close access ReviewRef
  try
    if (OldPic as text) = (NewPic as text) then set OldPic to ("Macintosh HD:Library:Desktop Pictures:Nature:Ladybug.jpg" as alias)
  end try
tell application "Finder" to set desktop picture to OldPic
tell application "TextEdit"
  activate
  open alias ReviewFile
  set b to bounds of window 1
  set item 3 of b to ((item 1 of b) + 800)
  set item 4 of b to ((item 2 of b) + 250)
  set bounds of window 1 to b
  open alias CurrentFile
set bounds of window 1 to {1,1,DispX,DispY}
end tell

tell application "System Events"
    tell process "Finder" to tell menu bar 1 to tell menu "Apple" to tell menu item "Dock" to tell menu 1 to get name of every menu item
        if result contains "Turn Hiding On" then keystroke "d" using {command down, option down}
    end tell
    quit
end CloseFile

on EndRoutine()
    global LastOp, filename, ReadPrefs, myPath, SeeQT, ReadQT
    PauseRoutine()
    if SeeQT = 1 then set WriteQT to "Yes (Show)"
    if SeeQT = 0 then set WriteQT to "No (Don't show)"
    set ReadPrefs to OverWrite(LastOp, filename, ReadPrefs, "Preferences", myPath)
    set ReadPrefs to OverWrite(ReadQT, WriteQT, ReadPrefs, "Preferences", myPath)
    CloseFile()
end EndRoutine