File created: 7-Mar-2024 22:42:36 **{DSK}<home>frank>il>notecards>system>NCPROGINT.;2**

changes to: (FNS NCP.ChangeLoc)

previous date: 18-Oct-2023 22:03:03 {DSK}<home>frank>il>notecards>system>NCPROGINT.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ NCPROGINTCOMS
       [
```

;;; Notefile creation and access

```
        (FNS NCP.CreateNoteFile NCP.OpenNoteFile NCP.OpenNoteFileP NCP.ListOfOpenNoteFiles NCP.CloseNoteFiles
             NCP.CheckpointNoteFiles NCP.AbortNoteFiles NCP.CompactNoteFile NCP.CompactNoteFileInPlace
             NCP.RepairNoteFile NCP.DeleteNoteFile NCP.NoteFileFromFileName NCP.FileNameFromNoteFile
             NCP.NoteFileMenu NCP.CheckInNoteFile NCP.CheckOutNoteFile NCP.LockFileName
             NCP.NumCardSlotsRemaining NCP.ExpandNoteFileIndex NCP.NoteFileAddProp NCP.NoteFileClosingP
             NCP.ForgetNoteFileName NCP.RememberNoteFileName NCP.ValidNoteFileP)
```

;;; Creating and accessing NoteCard types and substances.

```
        (FNS NCP.CardTypes NCP.CreateCardType NCP.DeleteCardType NCP.CreateCardTypeStub NCP.ChangeCardTypeFields
             NCP.CardTypeSuper NCP.CardTypeLinkDisplayMode NCP.CardTypeFn NCP.CardTypeVar NCP.CardTypeP
             NCP.CardTypeFnP NCP.CardTypeVarP NCP.CardTypeFns NCP.CardTypeVars NCP.CardTypeDisplayedInMenu
             NCP.IsSubTypeOfP NCP.TextBasedP NCP.SketchBasedP NCP.GraphBasedP NCP.AutoLoadCardType
             NCP.AddSpecialCard NCP.RemoveSpecialCard NCP.CardsOfTypes)
```

;;; Creating Notecards and fileboxes

```
        (FNS NCP.CreateCard NCP.CreateTextCard NCP.CreateFileBox NCP.CreateBrowserCard NCP.CreateSketchCard
             NCP.CreateGraphCard NCP.MakeDocument NCP.MakeLinkIndex NCP.NewCardP)
```

;;; Opening, closing, activating, display, etc.

```
        (FNS NCP.OpenCard NCP.CloseCards NCP.DisplayCard NCP.UndisplayCards NCP.CacheCards NCP.UncacheCards
             NCP.CardCachedP NCP.CardDisplayedP NCP.CardWindow NCP.WindowFromCard NCP.CardFromWindow
             NCP.CardFromTextStream)
```

;;; Accessing cards and boxes

```
        (FNS NCP.CardType NCP.ValidCardP NCP.CardTitle NCP.FileCards NCP.UnfileCards NCP.CardParents
             NCP.FileBoxChildren NCP.CardNeighbors NCP.GetLinks NCP.CardPropList NCP.CardProp NCP.CardAddProp
             NCP.CardDelProp NCP.CardSubstance NCP.CardRegion NCP.CardAddText NCP.ChangeLoc NCP.DeleteCards
             NCP.FileBoxP NCP.AllCards NCP.AllBoxes NCP.ContentsFileBox NCP.OrphansFileBox NCP.ToBeFiledFileBox
             NCP.NoteFileFromCard NCP.CardNoteFile NCP.SameCardP NCP.CoerceToCard NCP.DetermineDisplayRegion
             NCP.LockListOfCards NCP.GetCrossFileLinkDestCard NCP.CardBeingDeletedP)
```

;;; Collecting, copying, moving, deleting, cards

```
        (FNS NCP.CollectCards NCP.CopyCards NCP.MoveCards)
```

;;; Creating and accessing links

```
        (FNS NCP.CreateLink NCP.LocalGlobalLink NCP.GlobalGlobalLink NCP.GlobalLocalLink NCP.LocalLocalLink
             NCP.LinkDesc NCP.LinkDisplayMode NCP.LinkType NCP.LinkSource NCP.LinkDestination NCP.DeleteLinks
             NCP.ValidLinkP NCP.AllLinks NCP.SameLinkP NCP.LinkFromLinkIcon NCP.MakeLinkIcon NCP.MarkCardDirty
             NCP.LinkIconAttachedBitMap)
```

;;; Creating and accessing link labels.

```
        (FNS NCP.CreateLinkType NCP.DeleteLinkType NCP.RenameLinkType NCP.LinkTypes NCP.ReverseLinkTypes
             NCP.UserLinkTypes NCP.ValidLinkTypeP NCP.SystemLinkTypeP)
```

;;; Dealing with card parts dates.

```
        (FNS NCP.CardDates)
```

;;; Open events card

```
        (FNS NCP.GetOpenEventsCard NCP.GetCloseEventsCard NCP.MarkAsNotNeedingFiling)
```

;;; Functions for adding menu items

```
        (FNS NCP.AddSessionIconMenuItem NCP.RemoveSessionIconMenuItem NCP.RestoreSessionIconMenu
             NCP.AddNoteFileIconMenuItem NCP.RemoveNoteFileIconMenuItem NCP.RestoreNoteFileIconMenu
             NC.CreateSessionIconNoteFileMenuItem NCP.AddDefaultNoteFileIconMiddleButtonItems
             NCP.NoticedNoteFileNamesMenu NCP.AddTitleBarMenuItemsToType NCP.AddTitleBarMenuItemsToWindow
             NCP.BringUpSessionIcon NCP.SessionIconWindow)
```

;;; Miscellaneous

```
        (FNS NCP.TitleSearch NCP.PropSearch NCP.WhichCard NCP.WhichNoteFile NCP.SelectCards
             NCP.DocumentParameters NCP.NoteCardsParameters NCP.PrintMsg NCP.ClearMsg NCP.AskUser NCP.AskYesOrNo
             NCP.RegisterCardByName NCP.ListRegisteredCards NCP.LookupCardByName NCP.UnregisterName
             NCP.DisplayedCards NCP.CardUserDataProp NCP.NoteFileProp NCP.SetUpTitleBar
             NCP.AddNoteFileIconMiddleButtonItems NCP.NoteFileIconWindow NCP.CoerceToInterestedWindow
             NCP.SetGrayShade NCP.MakeTypeIconBitMapSet)
        (DECLARE%: DONTEVAL@LOAD (P (MOVD 'NCP.WhichCard 'NCP.WC T)
                                    (MOVD 'NCP.WhichNoteFile 'NCP.WNF T)))
```

;;; Handy internal functions

```
        (FNS NCP.ReportError NCP.ReportWarning NCP.LinkAnchorDesc NCP.GetTypeRecord
             NCP.AddLeftButtonTitleBarMenuItems NCP.AddMiddleButtonTitleBarMenuItems NCP.CoerceToLinkDisplayMode
             )
```

;;; Global variables.

```
        (GLOBALVARS NCP.ErrorBrkWhenFlg NCP.LinkDisplayModes NCP.TypeFnsAssocLst NCP.NoteCardTypeFns
               NCP.NoteCardTypeVars NC.MakeDocParameters NC.CardTypes NC.SubstanceTypes NC.SystemLinkLabels
               NC.FiledCardLinkLabel NC.SubBoxLinkLabel NC.SelectingCardsMenu NC.SelectingCardMenu
               NC.UCASESystemLinkLabels NC.SourceLinkLabel NC.NoteCardsParameters NCP.NoticedNoteFileNames
               NCP.GrayShade)
        [INITVARS (NCP.GrayShade GRAYSHADE)
               (NCP.NoticedNoteFileNames NIL)
               (NCP.DefaultLinkIconAttachedBitMapSize 17)
               (NCP.ErrorBrkWhenFlg NIL)
               (NCP.LinkDisplayModes '(Icon Title Label Both))
               (NCP.NoteCardTypeFns '(MakeFn EditFn QuitFn GetFn PutFn CopyFn MarkDirtyFn DirtyPFn
                                          CollectLinksFn DeleteLinksFn UpdateLinkIconsFn InsertLinkFn
                                          TranslateWindowPositionFn))
               (NCP.NoteCardTypeVars '(SuperType StubFlg FullDefinitionFile LinkDisplayMode DefaultWidth
                                           DefaultHeight LinkAnchorModesSupported DisplayedInMenuFlg
                                           LinkIconAttachedBitMap LeftButtonMenuItems MiddleButtonMenuItems]
```

;;; Following is for backward compatibility with 1.2

```
        (DECLARE%: DONTEVAL@LOAD (P (MOVD 'NCP.OpenCard 'NCP.BringUpCard T)
                                    (MOVD 'NCP.CacheCards 'NCP.ActivateCards T)
                                    (MOVD 'NCP.CardCachedP 'NCP.ActiveCardP T)
                                    (MOVD 'NCP.CardTypeFnP 'NCP.ValidCardTypeFn T)
                                    (MOVD 'NCP.CardTypeP 'NCP.ValidCardType T)
                                    (MOVD 'NCP.CardTypeVarP 'NCP.ValidCardTypeVar T)
                                    (MOVD 'NCP.CloseCards 'NCP.DeactivateCards T)
                                    (MOVD 'NCP.ValidCardP 'NCP.ValidCard T)
                                    (MOVD 'NCP.ContentsFileBox 'NCP.GetContentsFileBox T)
                                    (MOVD 'NCP.OrphansFileBox 'NCP.GetOrphansFileBox T)
                                    (MOVD 'NCP.ToBeFiledFileBox 'NCP.GetToBeFiledFileBox T)
                                    (MOVD 'NCP.LinkSource 'NCP.GetLinkSource T)
                                    (MOVD 'NCP.LinkDestination 'NCP.GetLinkDestination T)
                                    (MOVD 'NCP.CreateLinkType 'NCP.CreateLinkLabel T)
                                    (MOVD 'NCP.DeleteLinkType 'NCP.DeleteLinkLabel T)
                                    (MOVD 'NCP.RenameLinkType 'NCP.RenameLinkLabel T)
                                    (MOVD 'NCP.LinkTypes 'NCP.GetLinkLabels T)
                                    (MOVD 'NCP.UserLinkTypes 'NCP.GetUserLinkLabels T)
                                    (MOVD 'NCP.ReverseLinkTypes 'NCP.GetReverseLinkLabels T)
                                    (MOVD 'NCP.ValidLinkTypeP 'NCP.ValidLinkLabel T)
                                    (MOVD 'NCP.ValidLinkP 'NCP.ValidLink T)))
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              NCPROGINT)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA)
                      (NLAML)
                      (LAMA NCP.NoteFileProp NCP.CardUserDataProp NCP.PrintMsg NCP.PropSearch NCP.LinkType
                            NCP.LinkDisplayMode NCP.CardRegion NCP.CardSubstance NCP.CardProp NCP.CardTitle
                            NCP.CardTypeDisplayedInMenu]))
```

;;; Notefile creation and access

```
(DEFINEQ
```

(**NCP.CreateNoteFile**

```
  [LAMBDA (FileName QuietFlg)                                    (* rht%: " 2-Mar-87 21:58")
```

(* * Prog intface function for creating a notefile.)

```
        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (LET ((FileNameWithExt (NC.DatabaseFileName NIL NIL NIL NIL FileName)))
         (if (INFILEP FileNameWithExt)
             then (NCP.ReportError 'NCP.CreateNoteFile (CONCAT "NCP.CreateNoteFile" "Filename " FileNameWithExt
                                                              " already exists."))
                  NIL
          else (NC.CreateDatabaseFile FileNameWithExt NIL NIL NIL NIL NIL NIL QuietFlg)
               FileNameWithExt])
```

(NCP.OpenNoteFile

```
  [LAMBDA (NoteFileOrFileName Don'tCreateFlg Convertw/oConfirmFlg QuietFlg MenuPosition ReadOnlyFlg
                Don'tCreateInterfaceFlg)                          (* pmi%: "24-Nov-87 09:27")
```

```
        (* * Prog's intface version of opening a notefile.)

        (* * rht 7/7/86%: Now takes QuietFlg and MenuPosition arg. Takes either NoteFile object or file name.)

        (* * rht 7/16/86%: Added ReadOnlyFlg arg.)

        (* * rht 7/26/86%: Added Don'tCreateInterfaceFlg)

        (* * Fix to bug |#391:| Now calls NC.OpenNoteFile instead of NC.OpenDatabaseFile.)

        (* * pmi 5/20/87%: Removed HashArray argument in calls to NC.OpenNoteFile.)

        (* * pmi 11/24/87%: Deleted extra NIL in call to NC.OpenNoteFile.)

    (if (type? NoteFile NoteFileOrFileName)
        then (NC.OpenNoteFile NoteFileOrFileName NIL Don'tCreateFlg Convertw/oConfirmFlg NIL NIL
                    Don'tCreateInterfaceFlg NIL NIL NIL MenuPosition QuietFlg (AND ReadOnlyFlg 'INPUT)
                    NIL)
      else (LET ((FileNameWithExt (NC.DatabaseFileName "Name of NoteFile to open:" " -- " T NIL
                                          NoteFileOrFileName)))
                (AND FileNameWithExt (NC.OpenNoteFile FileNameWithExt NIL Don'tCreateFlg Convertw/oConfirmFlg NIL
                                             NIL Don'tCreateInterfaceFlg NIL NIL NIL MenuPosition QuietFlg
                                             (AND ReadOnlyFlg 'INPUT)
                                             NIL])
```

(NCP.OpenNoteFileP

```
  [LAMBDA (NoteFile)                                              (* Newman "13-Nov-86 17:28")
```

```
        (* * Non-nil if NoteFile is an open notefile.)

        (* * rht 9/19/86%: Fixed to return nil if notefile has no stream.)

        (* * dvn |11/13/86| Changed to call NC.NoteFileOpenP)

    (NC.NoteFileOpenP NoteFile])
```

(NCP.ListOfOpenNoteFiles

```
  [LAMBDA NIL                                                     (* rht%: "19-Sep-86 22:52")
```

```
        (* * Return list of all currently open notefiles.)

        (* * rht 9/19/86%: Fixed to return only notefiles currently open, not all that were ever open.)

    (for NoteFile in (NC.ListOfNoteFiles) when (NCP.OpenNoteFileP NoteFile) collect NoteFile])
```

(NCP.CloseNoteFiles

```
  [LAMBDA (NoteFilesOrT QuietFlg AutoConfirmFlg)                  (* Randy.Gobbel " 4-Mar-87 16:16")
```

```
        (* * Prog's intface function for closing a notefile.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 7/7/86%: Now takes list of notefiles. If arg is T, then close all open notefiles.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

        (* * 11/16/86%: Now calls NC.CloseNoteFile instead of NC.CloseDatabaseFile.)

        (* * rht 2/16/87%: Now takes AutoConfirmFlg and passes to NC.CloseNoteFile.)

    (for NoteFile in (if (EQ NoteFilesOrT T)
                         then (NCP.ListOfOpenNoteFiles)
                       else (MKLIST NoteFilesOrT))
       do (if (NOT (NCP.OpenNoteFileP NoteFile))
              then (NCP.ReportError 'NCP.CloseNoteFiles (CONCAT "Argument " NoteFile " is not a currently open
                                                               notefile."))
                   NIL
            else (NC.CloseNoteFile NoteFile NIL QuietFlg])
```

(NCP.CheckpointNoteFiles

```
[LAMBDA (NoteFilesOrT QuietFlg)                                    (* rht%: " 2-Mar-87 21:58")
```

(* * Checkpoint the given notefiles. If T then checkpoint all open notefiles.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (for NoteFile in (if (EQ NoteFilesOrT T)
                         then (NCP.ListOfOpenNoteFiles)
                       else (MKLIST NoteFilesOrT))
       do (if (NOT (NCP.OpenNoteFileP NoteFile))
              then (NCP.ReportError 'NCP.CheckpointNoteFiles (CONCAT "Argument " NoteFile " is not a currently
                                                                     open notefile."))
                   NIL
            else (NC.CheckpointDatabase NoteFile QuietFlg])
```

(NCP.AbortNoteFiles

```
[LAMBDA (NoteFilesOrT Don'tConfirmFlg QuietFlg)                    (* rht%: " 2-Mar-87 21:59")
```

(* * Abort the given notefiles. If T then abort all open notefiles.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (for NoteFile in (if (EQ NoteFilesOrT T)
                         then (NCP.ListOfOpenNoteFiles)
                       else (MKLIST NoteFilesOrT))
       do (if (NOT (NCP.OpenNoteFileP NoteFile))
              then (NCP.ReportError 'NCP.AbortNotefiles (CONCAT "Argument " NoteFile " is not a currently open
                                                                notefile."))
                   NIL
            else (NC.AbortSession NoteFile NIL Don'tConfirmFlg QuietFlg])
```

(NCP.CompactNoteFile

```
[LAMBDA (FromNoteFileOrFileName ToFileName InPlaceFlg)             (* rht%: " 7-Jul-86 19:23")
```

(* * Prog's intface function for compacting a notefile. FromNoteFile can be either a NoteFile or a file name.)

```
    (NC.CompactNoteFile FromNoteFileOrFileName ToFileName InPlaceFlg])
```

(NCP.CompactNoteFileInPlace

```
[LAMBDA (NoteFileOrFileName)                                       (* rht%: " 7-Jul-86 15:45")
```

(* * Prog's intface function for compacting a notefile in place.)

```
    (NCP.CompactNoteFile NoteFileOrFileName NIL T])
```

(NCP.RepairNoteFile

```
[LAMBDA (NoteFileOrFileName ReadSubstancesFlg)                     (* rht%: " 2-Mar-87 22:08")
```

(* * Prog's intface function for running inspect&repair on a notefile.)

(* * rht 7/17/86%: Now takes and passes ReadSubstancesFlg and calls NC.InspectAndRepairNoteFile instead of NC.ScavengerPhase1.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (if (NCP.OpenNoteFileP NoteFileOrFileName)
        then (NCP.ReportError 'NCP.RepairNoteFile (CONCAT "Can't inspect&repair an open notefile: "
                                                          NoteFileOrFileName))
      else (NC.InspectAndRepairNoteFile NoteFileOrFileName ReadSubstancesFlg])
```

(NCP.DeleteNoteFile

```
[LAMBDA (NoteFileOrFileName Don'tConfirmFlg QuietFlg)              (* pmi%: "31-Mar-87 18:31")
```

(* * Prog's intface function for deleting a notefile.)

(* * pmi 3/31/97%: Changed prompt from "Name of NoteFile to open:" to "Name of NoteFile to delete:")

```
    (if (type? NoteFile NoteFileOrFileName)
        then (NC.DeleteDatabaseFile NoteFileOrFileName NIL Don'tConfirmFlg QuietFlg)
      else (LET ((FileNameWithExt (NC.DatabaseFileName "Name of NoteFile to delete:" " -- " T NIL
                                                       NoteFileOrFileName)))
             (AND FileNameWithExt (NC.DeleteDatabaseFile FileNameWithExt NIL Don'tConfirmFlg QuietFlg])
```

(NCP.NoteFileFromFileName

```
[LAMBDA (FileName)                                                 (* rht%: " 7-Jul-86 19:34")
```

(* * Find notefile object for this filename, if any.)

```
    (NC.NoteFileFromFileName FileName])
```

(NCP.FileNameFromNoteFile

```
  [LAMBDA (NoteFile)                                        (* rht%: " 2-Mar-87 22:08")
```

(* * Grab filename off of this notefile object.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (if (NOT (type? NoteFile NoteFile))
        then (NCP.ReportError 'NCP.FileNameFromNoteFile (CONCAT "Arg to NCP.FileNameFromNoteFile is not a
                                                                notefile: " NoteFile))
      else (fetch (NoteFile FullFileName) of NoteFile])
```

(NCP.NoteFileMenu

```
  [LAMBDA (NoteFile)                                        (* rht%: "14-Jul-86 10:31")
```

(* * Fetch the notefile's menu.)

```
    (AND (type? NoteFile NoteFile)
         (fetch (NoteFile Menu) of NoteFile])
```

(NCP.CheckInNoteFile

```
  [LAMBDA (FromFile ToFile)                                 (* rht%: "19-Dec-84 20:22")
```

(* * Check lock file for ToFile. If none, then just copy FromFile to ToFile.
If there is one, then must be owned by us. If date of ToFile is more recent than date of lock file, then ask for user confirmation.)

```
    (PROG (LockFile User)
          (SETQ ToFile (NC.DatabaseFileName "Name of file to check in to: " "--" T T ToFile))
          [COND
             ((SETQ LockFile (INFILEP (NCP.LockFileName ToFile)))(* lock file exists.)
              (COND
                 ((EQ (GETFILEINFO LockFile 'LENGTH)
                      0)                                    (* Lock file is empty so delete it.)
                  (DELFILE LockFile))
                 ([NEQ (USERNAME NIL T)
                       (SETQ User (READ (SETQ LockFile (OR (OPENP LockFile 'INPUT)
                                                           (OPENSTREAM LockFile 'INPUT 'OLD]
                                                            (* someone else is playing with it.)
                  (PRIN1 (CONCAT "Can't check in because file was locked by " User " at " (GETFILEINFO
                                                                                           LockFile
                                                                                           'WRITEDATE)
                                 (CHARACTER 13)
                                 "To override, delete lock file and try again."
                                 (CHARACTER 13)))
                  (CLOSEF LockFile)
                  (RETURN NIL))
                 ((ILESSP (GETFILEINFO LockFile 'IWRITEDATE)
                          (GETFILEINFO ToFile 'IWRITEDATE))      (* Someone wrote the file since we locked it.)
                  (PRIN1 (CONCAT "Can't check in because file was locked by " User " at " (GETFILEINFO
                                                                                           LockFile
                                                                                           'WRITEDATE)
                                 " but written by someone at "
                                 (GETFILEINFO ToFile 'WRITEDATE)
                                 (CHARACTER 13)
                                 "To override, delete lock file and try again."
                                 (CHARACTER 13)))
                  (RETURN NIL))
                 (T                                         (* It's the lock file we wrote when checking out so just delete it.)
                    (CLOSEF LockFile)
                    (DELFILE LockFile]
          (SETQ FromFile (NC.DatabaseFileName "Name of file to check in from: " "--" T T FromFile))
          (PRIN1 (CONCAT "Copying " FromFile " to " ToFile " ... "))
          (COPYFILE FromFile ToFile)
          (PRIN1 (CONCAT "Done." (CHARACTER 13)))
          (RETURN (FULLNAME ToFile])
```

(NCP.CheckOutNoteFile

```
  [LAMBDA (FromFile ToFile)                                 (* Randy.Gobbel "18-Nov-86 15:42")
```

(* * Copy FromFile to ToFile unless FromFile is locked. Create a lock file in FromFile's directory.)

(* * rg 11/18/86%: Replaced call to obsolete VERSIONNUMBER fn with FILENAMEFIELD)

```
    (PROG (LockFile User)
          (SETQ FromFile (NC.DatabaseFileName "Name of file to check out: " "--" T T FromFile))
      LP  (SETQ LockFile (NCP.LockFileName FromFile))
          (COND
             [(INFILEP LockFile)                            (* lock file already exists.)
              (COND
                 ((EQ (GETFILEINFO LockFile 'LENGTH)
```

```
                    0)                                          (* Lock file is empty. Delete and start over.)
                 (DELFILE LockFile)
                 (GO LP))
                (T                                              (* someone else already is playing with it.)
                   [SETQ LockFile (OR (OPENP LockFile 'INPUT)
                                      (OPENSTREAM LockFile 'INPUT 'OLD]
                   (SETQ User (READ LockFile))
                   (PRIN1 (CONCAT "File is locked by: " User (CHARACTER 13)))
                   (CLOSEF LockFile)
                   (RETURN NIL]
             [(SETQ LockFile (OPENSTREAM LockFile 'OUTPUT))
              (COND
                 ((EQ (FILENAMEFIELD LockFile 'VERSION)
                      1)
                  (PRINT (USERNAME NIL T)
                         LockFile)
                  (CLOSEF LockFile))
                 (T                                             (* someone else created one before us.
                                                                Delete this one and try again.)
                   (DELFILE (CLOSEF LockFile))
                   (GO LP]
             (T                                                 (* something when wrong)
                (PRIN1 (CONCAT "Trouble in NC.CheckOutNoteFile." (CHARACTER 13)))
                (RETURN NIL)))
          (SETQ ToFile (NC.DatabaseFileName "Name of file to check out to: " "--" T T ToFile))
          (PRIN1 (CONCAT "Copying " FromFile " to " ToFile " ... "))
          (COPYFILE FromFile ToFile)
          (PRIN1 (CONCAT "Done." (CHARACTER 13)))
          (RETURN (FULLNAME ToFile])
```

(NCP.LockFileName

```
[LAMBDA (FileName)                                              (* rht%: "19-Dec-84 12:09")
                                                                (* returns the name of the lock file associated with FileName)
  (PACKFILENAME (LIST 'EXTENSION (PACK* (FILENAMEFIELD FileName 'EXTENSION)
                                        "LOCKFILE")
                      'VERSION 1 'BODY FileName])
```

(NCP.NumCardSlotsRemaining

```
[LAMBDA (NoteFile)                                              (* rht%: "24-May-87 00:04")

        (* * Return the number of card slots remaining in NoteFile. After they run out, it will have to have its index expanded.)

  (if (NCP.OpenNoteFileP NoteFile)
      then (DIFFERENCE (fetch (NoteFile HashArraySize) of NoteFile)
                       (NC.TotalCardsInNoteFile NoteFile))
    else (NCP.ReportError 'NCP.NumCardSlotsRemaining (CONCAT NoteFile " is not an open notefile."])
```

(NCP.ExpandNoteFileIndex

```
[LAMBDA (NoteFile NumNewSlots QuietFlg)                         (* rht%: "24-May-87 00:37")

        (* * Expand NoteFile's index in place after first checkpointing. Add room for NumNewSlots new slots.)

  (LET (WasOpenFlg)
       (if (SETQ WasOpenFlg (NCP.OpenNoteFileP NoteFile))
           then (NCP.CheckpointNoteFiles NoteFile QuietFlg)
         else (NCP.OpenNoteFile NoteFile T NIL QuietFlg NIL NIL T))
       (NC.ExpandIndexInPlace NoteFile (PLUS (fetch (NoteFile HashArraySize) of NoteFile)
                                             NumNewSlots)
              NIL NIL NIL QuietFlg)
       (OR WasOpenFlg (NCP.CloseNoteFiles NoteFile QuietFlg])
```

(NCP.NoteFileAddProp

```
[LAMBDA (NoteFile PROP NEWVALUE)                                (* pmi%: "23-Oct-87 15:02")

        (* * Prog. int. way into NC.NoteFileAddProp.)

  (NC.NoteFileAddProp NoteFile PROP NEWVALUE])
```

(NCP.NoteFileClosingP

```
[LAMBDA (DontCheckForAbortFlg)                                  (* DSJ%: " 7-Nov-87 00:28")

        (* * dsj |10/12/87.| This fn looks up the stack to see if the notefile's being closed.
        Not pretty, but there's no other way to tell. Unless DontCheckForAbortFlg, aborting the notefile counts as closing it.)

  (OR (RELSTK (STKPOS 'NC.CloseNoteFile))
      (AND (NOT DontCheckForAbortFlg)
           (RELSTK (STKPOS 'NC.AbortSession])
```

(NCP.ForgetNoteFileName

```
[LAMBDA (NoteFileOrFileName)                                    ; Edited 20-Oct-88 11:04 by RAR
  ;; Remove this notefile from the NoticedNoteFile list (menu).
```

```
    (LET [(noteFileObj (if (type? NoteFile NoteFileOrFileName)
                           then NoteFileOrFileName
                         else (NC.NoteFileFromFileName FullFileName]
         (if noteFileObj
             then (NC.RemoveNoteFileName NoteFileOrFileName)
                  (NCP.NoteFileProp noteFileObj 'UnNoticable T])
```

(NCP.RememberNoteFileName

```
  [LAMBDA (NoteFileOrFileName)                                    ; Edited 20-Oct-88 11:04 by RAR
    ;; Add this notefile to the NoticedNoteFile list (menu).
    (LET [(noteFileObj (if (type? NoteFile NoteFileOrFileName)
                           then NoteFileOrFileName
                         else (NC.NoteFileFromFileName FullFileName]
         (if noteFileObj
             then (NCP.NoteFileProp noteFileObj 'UnNoticable NIL)
                  (NC.NoticeNoteFileName NoteFileOrFileName])
```

(NCP.ValidNoteFileP

```
  [LAMBDA (noteFile)                                              ; Edited 24-Oct-88 13:43 by RAR
    ;; Is the notefile on the list of notefiles?
    (FMEMB noteFile (NC.ListOfNoteFiles])

)

;;; Creating and accessing NoteCard types and substances.

(DEFINEQ
```

(NCP.CardTypes

```
  [LAMBDA NIL                                                     (* rht%: "26-Oct-84 17:06")

          (* * Return list of all known card type names.)

    (NC.ListOfCardTypes])
```

(NCP.CreateCardType

```
  [LAMBDA (TypeName SuperType FnsAssocList VarsAssocList)         (* rht%: "16-Nov-86 22:46")

          (* * Make a new card type. If there is already a card type by that name, then print message and overwrite.)

          (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

          (* * rht 11/16/86%: Changed call to NCP.ReportWarning)

    (if (FMEMB TypeName (NCP.CardTypes))
        then (NCP.ReportWarning "NCP.CreateCardType" (CONCAT "Redefining NoteCard type: " TypeName)))
    (NC.AddCardType TypeName SuperType FnsAssocList VarsAssocList)
    TypeName])
```

(NCP.DeleteCardType

```
  [LAMBDA (TypeName DeleteSubTypesFlg)                            (* Randy.Gobbel "15-Sep-87 18:50")
    (NC.DeleteCardType TypeName DeleteSubTypesFlg])
```

(NCP.CreateCardTypeStub

```
  [LAMBDA (TypeName SuperType FullDefinitionFileName FnsAssocList VarsAssocList ListOfFILLMEFields)
                                                                  (* Randy.Gobbel "17-Dec-86 14:21")

          (* * Make a stub for a new card type. If there is already a card type by that name, then print message and overwrite.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

          (* * rg 12/17/86%: Added ListOfFILLMEFields argument.)

    (if (FMEMB TypeName (NCP.CardTypes))
        then (NCP.ReportWarning "NCP.CreateCardTypeStub" (CONCAT "Redefining NoteCard type: " TypeName)))
    (NC.AddCardTypeStub TypeName SuperType FullDefinitionFileName FnsAssocList VarsAssocList ListOfFILLMEFields)
    TypeName])
```

(NCP.ChangeCardTypeFields

```
  [LAMBDA (TypeName FnsAssocList VarsAssocList)                   (* rht%: " 2-Mar-87 22:08")

          (* * Change the given fields of TypeName.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (NCP.CardTypeP TypeName)
        then (NC.RecomputeCardType TypeName FnsAssocList VarsAssocList)
```

```
      else (NCP.ReportError 'NCP.ChangeCardTypeFields (CONCAT "Undefined card type: " TypeName])
```

(NCP.CardTypeSuper

```
  [LAMBDA (Type)                                                    (* rht%: " 2-Mar-87 21:59")

          (* * Return the super type for this type.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (NCP.CardTypeP Type)
        then (NC.GetCardTypeField SuperType Type)
      else (NCP.ReportError 'NCP.CardTypeSuper (CONCAT Type " is not a loaded NoteCard type."))
          NIL])
```

(NCP.CardTypeLinkDisplayMode

```
  [LAMBDA (Type)                                                    (* rht%: " 2-Mar-87 21:59")

          (* * Return the link display mode for this type.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (NCP.CardTypeP Type)
        then (NC.GetCardTypeField LinkDisplayMode Type)
      else (NCP.ReportError 'NCP.CardTypeLinkDisplayMode (CONCAT Type " is not a loaded NoteCard type."))
          NIL])
```

(NCP.CardTypeFn

```
  [LAMBDA (TypeName Fn)                                             ; Edited  3-Dec-87 19:00 by rht:

          (* * Return the function stored as the Fn for TypeName's record.)

          (* * rht 7/7/86%: Replaced NIL DEC in RECORDACCESS call with RECLOOK)

          (* * rht 8/25/86%: Changed RECORDACCESS to slightly less scuzzy call to NC.GetCardTypeField, the advantage is that
          the latter will force autoload if necessary.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (NCP.CardTypeP TypeName)
        then (if (NCP.ValidCardTypeFn Fn)
                 then (EVAL `(NC.GetCardTypeField ,Fn TypeName))
               else (NCP.ReportError 'NCP.CardTypeFn (CONCAT Fn " is not a kind of Fn for NoteCard types.")))
      else (NCP.ReportError 'NCP.CardTypeFn (CONCAT TypeName " is not a loaded NoteCard type."))
          NIL])
```

(NCP.CardTypeVar

```
  [LAMBDA (TypeName Var)                                            ; Edited  3-Dec-87 19:00 by rht:

          (* * Return the variable stored as the Var for TypeName's record.)

          (* * kirk 26Feb86 Replaced NIL DEC in RECORDACCESS call with RECLOOK)

          (* * rht 8/25/86%: Changed RECORDACCESS to slightly less scuzzy call to NC.GetCardTypeField, the advantage is that
          the latter will force autoload if necessary.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (NCP.CardTypeP TypeName)
        then (if (NCP.ValidCardTypeVar Var)
                 then (EVAL `(NC.GetCardTypeField ,Var TypeName))
               else (NCP.ReportError 'NCP.CardTypeVar (CONCAT Var " is not a kind of Var for NoteCard types.")))
      else (NCP.ReportError 'NCP.CardTypeVar (CONCAT TypeName " is not a loaded NoteCard type."))
          NIL])
```

(NCP.CardTypeP

```
  [LAMBDA (TypeName)                                                ; Edited 23-Feb-89 10:13 by krivacic

          (* * Returns non-nil if this TypeName is an existing NoteCard type.)

    (AND (GETHASH TypeName NC.CardTypes)
         TypeName])
```

(NCP.CardTypeFnP

```
  [LAMBDA (CardTypeFn)                                              (* rht%: " 7-Jul-86 21:26")

          (* * Returns non-nil if CardTypeFn is one of the Fn fields of the NoteCardType record.)

    (AND (FMEMB CardTypeFn NCP.NoteCardTypeFns)
         CardTypeFn])
```

(NCP.CardTypeVarP

```
[LAMBDA (CardTypeVar)                                    (* rht%: " 7-Jul-86 21:28")

          (* * Returns non-nil if CardTypeVar is one of the Var fields of the NoteCardType record.)

    (AND (FMEMB CardTypeVar NCP.NoteCardTypeVars)
         CardTypeVar])
```

(NCP.CardTypeFns

```
[LAMBDA NIL                                              (* rht%: " 7-Jul-86 20:58")

          (* * Returns list of the fns fields of the NoteCardType record.)

    NCP.NoteCardTypeFns])
```

(NCP.CardTypeVars

```
[LAMBDA NIL                                              (* rht%: " 7-Jul-86 20:59")

          (* * Returns list of the vars fields of the NoteCardType record.)

    NCP.NoteCardTypeVars])
```

(NCP.CardTypeDisplayedInMenu

```
[LAMBDA Args                                             ; Edited  3-Dec-87 19:00 by rht:

          (* * Expects one or two args%: CardType and optional new value for CardDisplayedInMenuFlg.
          Always returns the old value of CardDisplayedInMenuFlg for given card type.)

          (* * rht 7/15/86%: Changed to call NCP.ChangeCardTypeFields)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (LET (CardType OldVal)
         (if (OR (EQ Args 1)
                 (EQ Args 2))
             then (if (NCP.CardTypeP (SETQ CardType (ARG Args 1)))
                      then (PROG1 (SETQ OldVal (NCP.CardTypeVar CardType 'DisplayedInMenuFlg))
                                  [if (AND (EQ Args 2)
                                           (NEQ OldVal (EQ Args 2)))
                                      then (NCP.ChangeCardTypeFields CardType NIL
                                                                     `((DisplayedInMenuFlg ,(EQ Args 2])
                           else (NCP.ReportError 'NCP.CardTypeDisplayedInMenu (CONCAT CardType " is not a loaded NoteCard
                                                                                     type."))
                                NIL)
              else (NCP.ReportError 'NCP.CardTypeDisplayedInMenu "Improper number of args to
                                    NCP.CardTypeDisplayedInMenu.")
                   NIL])
```

(NCP.IsSubTypeOfP

```
[LAMBDA (SubTypeName SupposedSuperTypeName)              (* rht%: " 2-Mar-87 21:54")

          (* * Return non-nil if SubTypeName inherits somewhere up the hierarchy from SupposedSuperTypeName)

          (* * rht 8/19/86%: Fixed typo bug.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

          (* * rht 3/2/87%: No longer checks for valid card type, thus allowing autoloading if card type not currently loaded.)

    (NC.IsSubTypeOfP SubTypeName SupposedSuperTypeName])
```

(NCP.TextBasedP

```
[LAMBDA (CardOrCardType)                                 (* rht%: " 2-Mar-87 21:55")

          (* * This card or card type is a subtype of Text.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

          (* * rht 3/2/87%: No longer checks for valid card type, thus allowing autoloading if card type not currently loaded.)

    (LET ((CardType (if (NC.CardP CardOrCardType)
                        then (NCP.CardType CardOrCardType)
                      else CardOrCardType)))
         (NC.IsSubTypeOfP CardType 'Text])
```

(NCP.SketchBasedP

```
[LAMBDA (CardOrCardType)                                 (* rht%: " 2-Mar-87 21:55")

          (* * This card type is a subtype of Sketch.)

          (* * rht 10/15/86%: Fixed typo.)
```

```
          (* * rht 11/16/86%: Changed call to NCP.ReportError)

          (* * rht 3/2/87%: No longer checks for valid card type, thus allowing autoloading if card type not currently loaded.)

    (LET ((CardType (if (NC.CardP CardOrCardType)
                        then (NCP.CardType CardOrCardType)
                      else CardOrCardType)))
         (NC.IsSubTypeOfP CardType 'Sketch])
```

(NCP.GraphBasedP

```
  [LAMBDA (CardOrCardType)                                          (* rht%: " 2-Mar-87 21:54")

          (* * This card type is a subtype of Graph.)

          (* * rht 10/15/86%: Fixed typo.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

          (* * rht 3/2/87%: No longer checks for valid card type, thus allowing autoloading if card type not currently loaded.)

    (LET ((CardType (if (NC.CardP CardOrCardType)
                        then (NCP.CardType CardOrCardType)
                      else CardOrCardType)))
         (NC.IsSubTypeOfP CardType 'Graph])
```

(NCP.AutoLoadCardType

```
  [LAMBDA (TypeName)                                                (* rht%: "15-Jul-86 18:45")

          (* * Try to load the file containing TypeName looking in NOTECARDSDIRECTORIES.)

    (NC.AutoLoadCardType TypeName 'SuperType])
```

(NCP.AddSpecialCard

```
  [LAMBDA (Card)                                                    (* rht%: "23-Nov-86 13:33")

          (* * Adds this card to list of special cards hung off notefile's "SpecialCards" user data prop unless it's already there.)

    (LET ((NoteFile (NCP.CardNoteFile Card)))
         (if (for SpecialCard in (NCP.NoteFileProp NoteFile 'SpecialCards) never (NCP.SameCardP Card SpecialCard))
             then (NC.NoteFileAddProp NoteFile 'SpecialCards Card])
```

(NCP.RemoveSpecialCard

```
  [LAMBDA (Card)                                                    (* rht%: "20-Nov-86 15:07")

          (* * Removes this card from list of special cards hung off notefile's "SpecialCards" user data prop.)

    (NC.NoteFileDelProp (fetch (Card NoteFile) of Card)
           'SpecialCards Card (FUNCTION NC.SameCardP])
```

(NCP.CardsOfTypes

```
  [LAMBDA (CardsOrNoteFile Types)                                   (* pmi%: "18-Aug-87 11:17")

          (* * pmi 7/14/87%: First created to return all cards in CardsOrNoteFile which are of any of the types in Types.
          CardsOrNoteFile should be a card, a list of cards, or an open notefile.
          Types = NIL will return all cards in CardsOrNoteFile.)

          (* * pmi 8/18/87%: Modified to take an open notefile or a list of cards.)

    (LET (Cards)
         (if (NCP.OpenNoteFileP CardsOrNoteFile)
             then (SETQ Cards (NCP.AllCards CardsOrNoteFile))
           else (SETQ Cards (MKLIST CardsOrNoteFile)))
         (if Types
             then (SETQ Types (MKLIST Types))
                  (for Card in Cards when (FMEMB (NCP.CardType Card)
                                                 Types)
                     collect Card)
           else Cards])

)

;;; Creating Notecards and fileboxes

(DEFINEQ

(NCP.CreateCard

  [LAMBDA (Type NoteFile Title NoDisplayFlg Props ParentFileBoxes TypeSpecificArgs InterestedWindow
                RegionOrPosition)                                   ; Edited  5-Aug-88 15:57 by Trigg
    ;; Creates a new notecard with given type, title, props and parents.  Any of those args can be nil.  Type being NIL will cause user to be asked.
    ;; Makes a card with initially empty substance.
```

```
    ;; rht 11/20/84: Had to add a horrible kluge: if creating a document card in which embedded links may be copied, then need to have document card
    ;; visible on screen.  This is because ID is currently unattainable from just the Textstream --- need to have a window.  Until that is fixed, we
    ;; temporarily bring up the document card while it's being filled in.

    ;; rht 11/17/85: Updated to handle new card and notefile objects.

    ;; rht 7/10/86: Ripped out the above described document kludge.

    ;; rht 8/21/86: Now sticks (APPEND Props) on the card's proplist, i.e.  a top level copy.

    ;; rht 11/16/86: Changed call to NCP.ReportError

    ;; dsj 9/23/87: Added (NC.RetrievePropList Card) to APPEND to preserve properties set by user's makefn.

    ;; pmi 12/10/87: Added dsj's change;  see above comment.

    ;; pmi 2/23/88: Added InterestedWindow argument to passed on to NC.MakeNoteCard.

    ;; rht 8/5/88: Added RegionOrPosition arg and passed to NC.MakeNoteCard.

    (if (NCP.CardTypeP Type)
        then (LET (Card CardIdentifier)
                  (if (SETQ CardIdentifier (NC.MakeNoteCard Type NoteFile Title NoDisplayFlg TypeSpecificArgs NIL
                                                            InterestedWindow RegionOrPosition))
                      then (SETQ Card (if (WINDOWP CardIdentifier)
                                          then (NCP.CardFromWindow CardIdentifier)
                                        else CardIdentifier))
                           (if Props
                               then (NC.SetPropList Card (APPEND Props (NC.RetrievePropList Card)))
                                    (NC.SetPropListDirtyFlg Card T))
                           [for Box in (MKLIST ParentFileBoxes) do (if (NCP.FileBoxP Box)
                                                                       then (NCP.FileCards Card Box)
                                                                     else (NCP.ReportError 'NCP.CreateCard
                                                                            (CONCAT Box " not an existing
                                                                                   filebox."]
                           Card))
      else (NCP.ReportError 'NCP.CreateCard (CONCAT "Unknown card type: " Type))
           NIL])
```

(NCP.CreateTextCard

```
  [LAMBDA (NoteFile Title NoDisplayFlg Props ParentFileBoxes InterestedWindow RegionOrPosition)
                                                                          ; Edited  5-Aug-88 16:28 by Trigg
```

;;; Creates a new TEXT notecard with given title, props and parents. Makes a card with initially empty textstream.

```
    ;; rht 11/17/85: Updated to handle new card and notefile objects.

    ;; rht 8/5/88: Added TypeSpecificArgs, InterestedWindow, and RegionOrPosition args and passed to NCP.CreateCard.

    (NCP.CreateCard 'Text NoteFile Title NoDisplayFlg Props ParentFileBoxes NIL InterestedWindow RegionOrPosition
          ])
```

(NCP.CreateFileBox

```
  [LAMBDA (NoteFile Title NoDisplayFlg Props ChildCardsBoxes ParentFileBoxes InterestedWindow RegionOrPosition)
                                                                          ; Edited  5-Aug-88 16:27 by Trigg
```

;;; Creates a new Filebox with given title, props, children and parents. Does not display the card, but does give it an initial textstream

```
    ;; rht 11/17/85: Updated to handle new card and notefile objects.

    ;; rht 8/5/88: Added TypeSpecificArgs, InterestedWindow, and RegionOrPosition args and passed to NCP.CreateCard.

    (LET ((Card (NCP.CreateCard 'FileBox NoteFile Title NoDisplayFlg Props ParentFileBoxes NIL InterestedWindow
                      RegionOrPosition)))
         (NCP.FileCards ChildCardsBoxes Card)
        Card])
```

(NCP.CreateBrowserCard

```
  [LAMBDA (NoteFile Title ParamList NoDisplayFlg Props ParentFileBoxes InterestedWindow RegionOrPosition)
                                                                          ; Edited  5-Aug-88 16:02 by Trigg
```

```
    ;; Creates a new browser notecard with given type, title, props, parents, starting ID and link labels.

    ;; rht 11/17/85: Updated to handle new card and notefile objects.

    ;; rht 8/21/86: Changed call from NCP.ValidLinkType to NCP.ValidLinkTypeP.

    ;; rht 11/16/86: Changed call to NCP.ReportError

    ;; rht 3/7/87: Now properly accepts link types with '_' prefixes.

    ;; pmi 2/24/88: Now passes NIL as NewParamList if the user didn't specify any parameters.  Also added InterestedWindow argument.

    ;; rht 8/5/88: Added RegionOrPosition arg and passed to NCP.CreateCard.

    (LET (ValidLinkTypes LinkTypes NewParamList)
         [SETQ ValidLinkTypes (for LinkType in (SETQ LinkTypes (LISTGET ParamList 'LINKTYPES))
                                 join (COND
                                         ((EQ LinkType 'ALL)
                                          (NCP.LinkTypes NoteFile))
                                         ((EQ LinkType '_ALL)
                                          (NCP.ReverseLinkTypes NoteFile))
                                         ((OR (NCP.ValidLinkTypeP LinkType NoteFile)
                                              (AND (STREQUAL (SUBSTRING LinkType 1 1)
                                                             "_")
```

```
                                                (NCP.ValidLinkTypeP (SUBATOM LinkType 2)
                                                       NoteFile)))
                                          (LIST LinkType))
                                         (T (NCP.ReportError 'NCP.CreateBrowserCard (CONCAT LinkType " not a valid
                                                                                           link type."))
                                            NIL]
        (SETQ ValidLinkTypes (INTERSECTION ValidLinkTypes ValidLinkTypes))
        (if (AND LinkTypes (NULL ValidLinkTypes))
            then NIL
          else                                                  ; Make a copy of the user's param list since she may not want it
                                                                ; to get rplaca'd.
               (if ParamList
                   then (SETQ NewParamList (COPY ParamList))
                        (LISTPUT NewParamList 'LINKTYPES ValidLinkTypes)
                 else (SETQ NewParamList NIL))
               (NCP.CreateCard 'Browser NoteFile Title NoDisplayFlg Props ParentFileBoxes NewParamList
                      InterestedWindow RegionOrPosition])
```

(NCP.CreateSketchCard

```
  [LAMBDA (NoteFile Title NoDisplayFlg Props ParentFileBoxes InterestedWindow RegionOrPosition)
                                                                ; Edited  5-Aug-88 16:27 by Trigg
```

;;; Creates a new SKETCH/MAP notecard with given title, props and parents.

;; rht 11/17/85: Updated to handle new card and notefile objects.

;; rht 8/5/88: Added InterestedWindow and RegionOrPosition args and passed to NCP.CreateCard.

```
(NCP.CreateCard 'Sketch NoteFile Title NoDisplayFlg Props ParentFileBoxes NIL InterestedWindow
       RegionOrPosition])
```

(NCP.CreateGraphCard

```
  [LAMBDA (NoteFile Title NoDisplayFlg Props ParentFileBoxes InterestedWindow RegionOrPosition)
                                                                ; Edited  5-Aug-88 16:05 by Trigg
```

;;; Creates a new GRAPH notecard with given title, props and parents.

;; rht 11/17/85: Updated to handle new card and notefile objects.

;; rht 8/5/88: Added InterestedWindow and RegionOrPosition args and passed to NCP.CreateCard.

```
(NCP.CreateCard 'Graph NoteFile Title NoDisplayFlg Props ParentFileBoxes NIL InterestedWindow
       RegionOrPosition])
```

(NCP.MakeDocument

```
  [LAMBDA (NoteFile RootCard ParamProps NoDisplayFlg Props ParentFileBoxes InterestedWindow RegionOrPosition)
                                                                ; Edited  5-Aug-88 16:30 by Trigg
```

;; Do a MakeDocument starting from RootCard according to parameters in ParamProps if non-nil. Otherwise use the default parameters. Note that ;; ParamProps are *only* used for the duration of this MakeDocument and do not affect the default parameter values.

;; rht 11/17/85: Updated to handle new card and notefile objects.

;; rht 8/25/86: Now passes non-nil QuietFlg to NCP.UncacheCards.

;; rht 11/16/86: Changed call to NCP.ReportError

;; pmi 2/26/88: Now passes ParamProps on down to NC.MakeDocument, which will handle fetching the existing parameters and restoring the ;; original ones. Also added InterestedWindow argument to be passed on to NCP.CreateCard.

;; rht 8/5/88: Added RegionOrPosition arg and passed to NCP.CreateCard.

```
(LET (CurParams DocCard WasActive)
     (if (NC.ValidCardP RootCard)
         then (AND (NOT (SETQ WasActive (NCP.CardCachedP RootCard)))
                   (NCP.CacheCards RootCard))
              ;; (if ParamProps then (SETQ CurParams (NCP.DocumentParameters ParamProps)))
              (SETQ DocCard (NCP.CreateCard 'Document NoteFile NIL NoDisplayFlg Props ParentFileBoxes
                                 (LIST RootCard ParamProps)
                                 InterestedWindow RegionOrPosition))
              ;; (if ParamProps then (SETPROPLIST 'NC.MakeDocParameters CurParams))
              (AND (NOT WasActive)
                   (NCP.UncacheCards RootCard T))
              DocCard
       else (NCP.ReportError 'NCP.MakeDocument (CONCAT RootCard " not a valid card or filebox."))
            NIL])
```

(NCP.MakeLinkIndex

```
  [LAMBDA (NoteFile LinkTypes BackPointersP NoDisplayFlg Props ParentFileBoxes InterestedWindow RegionOrPosition
                )                                               ; Edited  5-Aug-88 16:49 by Trigg
```

;;; Do a MakeLinkIndex on LinkTypes inserting backpointers according to BackPointersP.

;; rht 11/17/85: Updated to handle new card and notefile objects.

;; rht 11/16/86: Changed call to NCP.ReportError

;; rht 8/5/88: Added RegionOrPosition and InterestedWindow args and passed to NCP.CreateCard.

```
(LET [(ValidLinkTypes (for LinkType in (MKLIST LinkTypes) join (COND
```

```
                                                                ((EQ LinkType 'ALL)
                                                                 (NCP.LinkTypes NoteFile))
                                                                ((EQ LinkType '_ALL)
                                                                 (NCP.ReverseLinkTypes NoteFile))
                                                                ((NOT (NCP.ValidLinkType LinkType NoteFile))
                                                                 (NCP.ReportError 'NCP.MakeLinkIndex
                                                                        (CONCAT LinkType " not a valid link
                                                                               label."))
                                                                 NIL)
                                                                (T (LIST LinkType]
        (SETQ ValidLinkTypes (INTERSECTION ValidLinkTypes ValidLinkTypes))
        (if (AND LinkTypes (NULL ValidLinkTypes))
            then NIL
          else (NCP.CreateCard 'LinkIndex NoteFile NIL NoDisplayFlg Props ParentFileBoxes (LIST ValidLinkTypes
                                                                                                BackPointersP)
                       InterestedWindow RegionOrPosition])
```

(NCP.NewCardP

```
  [LAMBDA (Card)                                                (* pmi%: "21-Oct-87 15:54")

          (* * pmi 10/21/87%: First created. Allows prog int users to tell if a card has been saved yet.)

    (NC.FetchNewCardFlg Card])

)
```

;;; Opening, closing, activating, display, etc.

```
(DEFINEQ
```

(NCP.OpenCard

```
  [LAMBDA (Card Region/Position TypeSpecificArgs ReadOnly)       ; Edited 22-Aug-88 14:44 by Burwell
```

;;; Cache this card, if necessary, and display on the screen.

;; rht 11/16/86: Changed call to NCP.ReportError

;; rht 8/8/88: Added check that old proc's window is open before giving it the tty.

;; rht 8/22/88: Fixed typo in above fix.

```
    (if (NC.ValidCardP Card)
        then [LET ((OldProc (TTY.PROCESS)))
                  (PROG1 (NC.EditNoteCard Card (OR ReadOnly (fetch (NoteFile ReadOnlyFlg)
                                                              of (fetch (Card NoteFile) of Card)))
                                 Region/Position TypeSpecificArgs)
                         (AND (PROCESSP OldProc)
                              (OPENWP (PROCESSPROP OldProc 'WINDOW))
                              (TTY.PROCESS OldProc)))]
      else (NCP.ReportError 'NCP.OpenCard (CONCAT Card " not an existing card or box."])
```

(NCP.CloseCards

```
  [LAMBDA (Cards QuietFlg)                                       ; Edited 22-Aug-88 14:43 by Burwell
```

;; Uncache and undisplay any active cards in Cards

;; rht 11/16/86: Changed call to NCP.ReportError

;; rht 3/9/87: Fixed so that wouldn't try to get PROCESS windowprop from NIL Win.

;; rg 3/9/87 fixed args to NC.QuitCard ; added NC.ProtectedSessionOperation wrapper

;; rg 4/2/87 changed NC.ProtectedSessionOperation to NCP.WithLockedCards ; added NC.IfAllCardsFree wrapper

;; rg 8/4/88 Changed CANCELLED to DON'T

;; rht 8/8/88: Added check that old proc's window is open before giving it the tty.

;; rht 8/22/88: Fixed typo in above fix.

```
    (NCP.WithLockedCards (NC.IfAllCardsFree
                          (NC.LockListOfCards (MKLIST Cards)
                                "Close Cards")
                          (for Card in (MKLIST Cards) bind Win (OldProc _ (TTY.PROCESS))
                             do (if (NOT (NC.ValidCardP Card))
                                    then (NCP.ReportError "NCP.CloseCards" (CONCAT Card " not an existing card or
                                                                                   filebox."))
                                  elseif (AND (NCP.CardCachedP Card)
                                              (NEQ (NC.QuitCard Card T NIL NIL NIL NIL QuietFlg)
                                                   'DON'T)
                                              (SETQ Win (NC.FetchWindow Card)))
                                    then (bind [Process _ (AND Win (WINDOWPROP Win 'PROCESS]
                                            until (OR (NULL Process)
                                                      (PROCESS.FINISHEDP Process))
                                            do (BLOCK)))
                             finally (AND (PROCESSP OldProc)
                                          (OPENWP (PROCESSPROP OldProc 'WINDOW))
                                          (TTY.PROCESS OldProc))
                                    (RETURN Card])
```

(NCP.DisplayCard

```
[LAMBDA (Card Region/Position TypeSpecificArgs ReadOnly)        ; Edited 22-Aug-88 14:44 by Burwell
```

;;; display Card on the screen.

;; rht 11/16/86: Changed call to NCP.ReportError

;; rg 11/4/87 added ReadOnly

;; rht 8/8/88: Added check that old proc's window is open before giving it the tty.

;; rht 8/22/88: Fixed typo in above fix.

```
(if (NC.ValidCardP Card)
    then (if (NCP.CardCachedP Card)
             then [LET ((OldProc (TTY.PROCESS)))
                       (PROG1 (NC.EditNoteCard Card (OR ReadOnly (fetch (NoteFile ReadOnlyFlg)
                                                                    of (fetch (Card NoteFile) of Card)))
                                                   Region/Position TypeSpecificArgs)
                              (AND (PROCESSP OldProc)
                                   (OPENWP (PROCESSPROP OldProc 'WINDOW))
                                   (TTY.PROCESS OldProc)))]
           else (NCP.ReportError 'NCP.DisplayCard (CONCAT Card " must be cached before displayed:
                                                          NCP.DisplayCard.")))
  else (NCP.ReportError 'NCP.DisplayCard (CONCAT Card " not an existing card or box."])
```

(NCP.UndisplayCards

```
[LAMBDA (Cards QuietFlg WriteChangesFlg)                        ; Edited 22-Aug-88 14:44 by Burwell
```

;;; If card is valid and displayed, then undisplay it but leave it cached. If WriteChangesFlg is non-nil, then save changes to the file, otherwise saving will
;;; wait until card is uncached.

;; rht 11/16/86: Changed call to NCP.ReportError

;; rht 8/8/88: Added check that old proc's window is open before giving it the tty.

;; rht 8/22/88: Fixed typo in above fix.

```
(for Card in (MKLIST Cards) do [COND
                                ((NOT (NC.ValidCardP Card))
                                 (NCP.ReportError 'NCP.UndisplayCards (CONCAT Card " not an existing card or
                                                                              filebox.")))
                                ((NOT (NCP.CardDisplayedP Card))
                                 (NCP.PrintMsg NIL T Card " already undisplayed: NCP.UndisplayCards"))
                                (T (LET ((OldProc (TTY.PROCESS)))
                                        (PROG1 (NC.QuitCard Card T (NOT WriteChangesFlg)
                                                            NIL NIL NIL QuietFlg T)
                                               (AND (PROCESSP OldProc)
                                                    (OPENWP (PROCESSPROP OldProc 'WINDOW))
                                                    (TTY.PROCESS OldProc)))]
   finally (RETURN Card])
```

(NCP.CacheCards

```
[LAMBDA (Cards)                                                 (* rht%: " 2-Mar-87 22:01")
```

(* * Cache all the info for any inactive cards in Cards)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
(for Card in (MKLIST Cards) do (COND
                                ((NOT (NC.ValidCardP Card))
                                 (NCP.ReportError 'NCP.CacheCards (CONCAT Card " not an existing card or
                                                                          filebox.")))
                                ((NCP.CardCachedP Card)
                                 (NCP.PrintMsg NIL T Card " already cached: NCP.CacheCards."))
                                (T (NC.GetNoteCard Card)))
   finally (RETURN Card])
```

(NCP.UncacheCards

```
[LAMBDA (Cards QuietFlg)                                        (* Randy.Gobbel " 4-Mar-87 14:43")
```

(* * Uncache and undisplay any active cards in Cards)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
(for Card in (MKLIST Cards) do (COND
                                ((NOT (NC.ValidCardP Card))
                                 (NCP.ReportError 'NCP.UncacheCards (CONCAT Card " not an existing card or
                                                                            filebox.")))
                                ((NOT (NCP.CardCachedP Card))
                                 (NCP.ReportError 'NCP.UncacheCards (CONCAT Card " must be cached before can
                                                                            uncache.")))
                                ((NCP.CardDisplayedP Card)
                                 (NCP.ReportError 'NCP.UncacheCards (CONCAT Card " must be undisplayed before
                                                                            can uncache.")))
                                (T (NC.QuitCard Card NIL NIL NIL NIL NIL QuietFlg])
```

(NCP.CardCachedP

```
  [LAMBDA (Card)                                          (* rht%: "10-Jul-86 21:57")

          (* * Return non-nil if card is cached.)

    (NC.ActiveCardP Card])
```

(NCP.CardDisplayedP

```
  [LAMBDA (Card)                                          (* rht%: "10-Jul-86 22:08")

          (* * Return non-nil if card is currently displayed on screen.)

    (AND (NCP.CardCachedP Card)
         (NCP.WindowFromCard Card])
```

(NCP.CardWindow

```
  [LAMBDA (Card)                                          (* rht%: "18-Nov-85 17:43")

          (* * Returns T if card corresponding to Card is currently on the screen, i.e.
          has an active window.)

          (* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

    (AND (NC.ValidCardP Card)
         (NC.FetchWindow Card])
```

(NCP.WindowFromCard

```
  [LAMBDA (Card)                                          (* rht%: "11-Jul-86 10:21")

          (* * Returns T if card corresponding to Card is currently on the screen, i.e.
          has an active window.)

    (NCP.CardWindow Card])
```

(NCP.CardFromWindow

```
  [LAMBDA (Win)                                           (* rht%: "10-Jul-86 22:10")

          (* * Return the card corresponding to Win, if Win is a notecard window.)

    (NC.CoerceToCard Win])
```

(NCP.CardFromTextStream

```
  [LAMBDA (TextStream)                                    (* rht%: "12-Jul-86 14:55")

          (* * Return the card corresponding to TextStream, if TextStream is a text card's textstream.)

    (NC.CoerceToCard TextStream])

)

;;; Accessing cards and boxes

(DEFINEQ
```

(NCP.CardType

```
  [LAMBDA (Card)                                          (* rht%: "18-Nov-85 01:23")

          (* * Return the type of ID or NIL if no such ID.)

          (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

    (AND (NC.ValidCardP Card)
         (NC.RetrieveType Card])
```

(NCP.ValidCardP

```
  [LAMBDA (Card)                                          (* pmi%: "10-Dec-87 11:34")

          (* * Non-nil if ID corresponds to extant card in current notefile. Returns type of card or nil.)

          (* * pmi 12/10/87%: Changed to call NC.ValidCardP rather than NC.CardType
          (how odd!) as a check for validity. Also now returns the card if successful.)

    (if (NC.ValidCardP Card)
        then Card])
```

(NCP.CardTitle

```
  [LAMBDA Args                                            (* rht%: " 2-Mar-87 22:01")

          (* * Expects one or two args, the Card and an optional new title.
          If the latter is present then change the title of ID. In any case, return the old title.)
```

```
        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (LET (Card)
         (COND
            ((AND (NEQ Args 1)
                  (NEQ Args 2))
             (NCP.ReportError 'NCP.CardTitle "Improper number of arguments to NCP.CardBoxTitle.")
             NIL)
            [(NC.ValidCardP (SETQ Card (ARG Args 1)))
             (PROG1 (NC.RetrieveTitle Card)
                 (if (EQ Args 2)
                     then (NC.AssignTitle Card NIL (ARG Args 2))))]
            (T (NCP.ReportError 'NCP.CardTitle (CONCAT Card " not an existing card or box."))
               NIL])
```

(NCP.FileCards

```
  [LAMBDA (CardBoxList BoxList)                                    (* rht%: " 2-Mar-87 22:01")

        (* * File every card or box in CardBoxList in every box in BoxList.
        Either arg can be an atom or list. Check for cycles.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (SETQ CardBoxList (for Card in (MKLIST CardBoxList) unless (if (NOT (NC.ValidCardP Card))
                                                                   then (NCP.ReportError 'NCP.FileCards
                                                                          (CONCAT Card " not an existing card
                                                                                 or box.  Can't file."))
                                                               T)
                          collect Card))
    (SETQ BoxList (for Card in (MKLIST BoxList) unless (if (NOT (NCP.FileBoxP Card))
                                                           then (NCP.ReportError 'NCP.FileCards (CONCAT Card " not
                                                                                                        an existing
                                                                                                        filebox.
                                                                                                        Can't be
                                                                                                        filed into.")
                                                                )
                                                       T)
                      collect Card))
    (if (AND BoxList CardBoxList)
        then (for Box in BoxList when (NC.FileBoxCollectChildren NIL Box CardBoxList T) collect Box)
      else NIL])
```

(NCP.UnfileCards

```
  [LAMBDA (CardBoxList BoxList)                                    (* rht%: " 2-Mar-87 22:01")

        (* * Unfile every card or box in CardBoxList from every box in BoxList.
        Either arg can be a litatom ID. Either can also be the litatom ALL.
        If CardBoxList is ALL then clear out all children from every element of BoxList.
        If BoxList is ALL then unlink all parents of every element of CardBoxList.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (NEQ CardBoxList 'ALL)
        then (SETQ CardBoxList (for Card in (MKLIST CardBoxList) unless (if (NOT (NC.ValidCardP Card))
                                                                            then (NCP.ReportError 'NCP.UnfileCards
                                                                                   (CONCAT Card " not an
                                                                                          existing card or
                                                                                          box.  Can't
                                                                                          unfile."))
                                                                        T)
                                   collect Card)))
    (if (NEQ BoxList 'ALL)
        then (SETQ BoxList (for Card in (MKLIST BoxList) unless (if (NOT (NCP.FileBoxP Card))
                                                                    then (NCP.ReportError 'NCP.UnfileCards
                                                                           (CONCAT Card " not an existing
                                                                                  filebox."))
                                                                T)
                           collect Card)))
    [if (NEQ CardBoxList 'ALL)
        then [if (EQ BoxList 'ALL)
                 then                                              (* Unfile every element of CardBoxList from all its parents.)
                      [for Card in CardBoxList do (NCP.DeleteLinks (NCP.GetLinks NIL Card (LIST NC.FiledCardLinkLabel
                                                                                               NC.SubBoxLinkLabel]
             else                                                  (* Unfile every element of CardBoxList from a selection of its
                                                                   parents.)
                  (for Card in CardBoxList do (for Box in BoxList do (NCP.DeleteLinks (NCP.GetLinks Box Card
                                                                                                    (LIST
                                                                                                      NC.FiledCardLinkLabel
```

```
                                                                                                    NC.SubBoxLinkLabel
                                                                                                     ]
      else (if (EQ BoxList 'ALL)
               then (SETQ BoxList (NCP.AllBoxes)))
           (for Box in BoxList do (NCP.DeleteLinks (NCP.GetLinks Box NIL (LIST NC.FiledCardLinkLabel
                                                                              NC.SubBoxLinkLabel]
    (AND CardBoxList BoxList])
```

(NCP.CardParents

```
[LAMBDA (Card)                                                          (* rht%: " 2-Mar-87 22:10")
```

(* * Return the list of fileboxes in which Card has been filed.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (if (NC.ValidCardP Card)
        then (for Link in (NCP.GetLinks NIL Card (LIST NC.FiledCardLinkLabel NC.SubBoxLinkLabel))
                collect (fetch (Link SourceCard) of Link))
      else (NCP.ReportError 'NCP.CardParents (CONCAT Card " is not an existing card or filebox."])
```

(NCP.FileBoxChildren

```
[LAMBDA (Box)                                                           (* rht%: " 2-Mar-87 22:10")
```

(* * Return the list of children of Box in proper order.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (COND
       ((NCP.FileBoxP Box)
        (for Link in (NCP.GetLinks Box NIL (LIST NC.FiledCardLinkLabel NC.SubBoxLinkLabel))
           collect (fetch (Link DestinationCard) of Link)))
       (T (NCP.ReportError 'NCP.FileBoxChildren (CONCAT Box " is not an existing filebox."])
```

(NCP.CardNeighbors

```
[LAMBDA (Cards LinkTypes FollowCrossFileLinksFlg)                       (* rht%: " 6-Jun-87 17:41")
```

(* * Return a list of cards one link from Card in the forward direction having some link of type in LinkTypes.
LinkTypes can contain atoms starting with "_" in which case it's a backward link.
LinkTypes can contain either or both of ANY and _ANY, meaning to include any forward or backward links, respectively.)

(* * rht 6/6/87%: Added FollowCrossFileLinksFlg; if non-nil, then try to follow cross-file links into remote notefiles, else ignore them.)

```
    (if (NULL LinkTypes)
        then [for Link in (NCP.GetLinks Cards) join (LET ((DestCard (fetch (Link DestinationCard) of Link)))
                                                      (MKLIST (if (NC.CrossFileLinkCardP DestCard)
                                                                  then (if FollowCrossFileLinksFlg
                                                                           then (NC.GetCrossFileLinkDestCard
                                                                                    DestCard)
                                                                         else NIL)
                                                                else DestCard]
      else (LET (ForwardLinkTypes BackwardLinkTypes)
                (for LinkType in (MKLIST LinkTypes) do (if (EQ (NTHCHAR LinkType 1)
                                                               '_)
                                                           then (pushnew BackwardLinkTypes (SUBATOM LinkType 2))
                                                         else (pushnew ForwardLinkTypes LinkType)))
                (UNION [AND ForwardLinkTypes (for Link in (NCP.GetLinks Cards NIL (if (FMEMB 'ANY ForwardLinkTypes)
                                                                                      then NIL
                                                                                    else ForwardLinkTypes))
                                                    join (LET ((DestCard (fetch (Link DestinationCard) of Link)))
                                                              (MKLIST (if (NC.CrossFileLinkCardP DestCard)
                                                                          then (if FollowCrossFileLinksFlg
                                                                                   then (NC.GetCrossFileLinkDestCard
                                                                                            DestCard)
                                                                                 else NIL)
                                                                        else DestCard]
                       (AND BackwardLinkTypes (for Link in (NCP.GetLinks NIL Cards (if (FMEMB 'ANY BackwardLinkTypes
                                                                                              )
                                                                                       then NIL
                                                                                     else BackwardLinkTypes))
                                                   join (LET ((SrcCard (fetch (Link SourceCard) of Link)))
                                                             (MKLIST (if (NC.CrossFileLinkCardP SrcCard)
                                                                         then (if FollowCrossFileLinksFlg
                                                                                  then (NC.GetCrossFileLinkDestCard
                                                                                           SrcCard)
                                                                                else NIL)
                                                                       else SrcCard])
```

(NCP.GetLinks

```
[LAMBDA (Cards DestinationCards Labels NoteFile)                        (* rht%: "29-May-87 16:48")
```

```
        (* * Returns a list of all links from Cards to DestinationCards whose link label is one of Labels.
        Labels can be nil, in which case all such links are returned. Cards and DestinationCards can each be atomic.
        Each can also be nil. For example, if DestinationCards is nil, then all links pointing from Cards to anywhere with given labels
        are returned. Note that if both Cards and DestinationCards are nil, then will return all links whose label is one of Labels.
        If all three args are nil, then return all links in the current notefile.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 8/29/86%: Now blocks in loops and checks whether links cached before retrieving.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

        (* * rht&pmi 5/29/87%: No longer has to uncache links because of change to NC.RetrieveFromLinks and
        NC.RetrieveToLinks.)

    (LET (ValidCards ValidDestinationCards)
         (SETQ Labels (MKLIST Labels))
         (SETQ ValidCards (for Card in (MKLIST Cards) eachtime (BLOCK) unless (COND
                                                                                 ((NOT (NC.ValidCardP Card))
                                                                                  (NCP.ReportError 'NCP.GetLinks
                                                                                         (CONCAT Card " not an
                                                                                                existing card or
                                                                                                box."))
                                                                                  T))
                                 collect Card))
         (SETQ ValidDestinationCards (for Card in (MKLIST DestinationCards) eachtime (BLOCK)
                                        unless (COND
                                                  ((NOT (NC.ValidCardP Card))
                                                   (NCP.ReportError 'NCP.GetLinks (CONCAT Card " not an existing
                                                                                         card or box."))
                                                   T))
                                        collect Card))
         (COND
            (Cards (for Card in ValidCards eachtime (BLOCK)
                      join (for Link in (NC.RetrieveToLinks Card) when (COND
                                                                          (DestinationCards
                                                                           (FMEMB (fetch (Link DestinationCard)
                                                                                     of Link)
                                                                                  ValidDestinationCards))
                                                                          (T T))
                                 when (COND
                                         (Labels (FMEMB (fetch (Link Label) of Link)
                                                        Labels))
                                         (T T))
                                 collect Link)))
            (DestinationCards (for Card in ValidDestinationCards eachtime (BLOCK)
                                 join (for Link in (NC.RetrieveFromLinks Card)
                                         when (COND
                                                 (Labels (FMEMB (fetch (Link Label) of Link)
                                                                Labels))
                                                 (T T))
                                         collect Link)))
            (T (NCP.MapLinks NoteFile (FUNCTION PROG1)
                       (FUNCTION (LAMBDA (Link)
                                    (if Labels
                                        then (FMEMB (fetch (Link Label) of Link)
                                                    Labels)
                                      else T])
```

(NCP.CardPropList

```
  [LAMBDA (Card)                                                    (* rht%: "18-Nov-85 01:45")
```

```
        (* * Return the ID's property list)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

    (NC.RetrievePropList Card])
```

(NCP.CardProp

```
  [LAMBDA Args                                                      (* rht%: " 2-Mar-87 22:02")
```

```
        (* * Expects two or three arguments%: ID, Property, and optional new value.
        Returns the old value. Assigns the new value if given. Semantics are just like WINDOWPROP.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (LET (Card PropList)
         (COND
            ((AND (NEQ Args 2)
                  (NEQ Args 3))
             (NCP.ReportError 'NCP.CardProp (CONCAT "Improper number of args: " Args))
             NIL)
```

```
            [(NC.ValidCardP (SETQ Card (ARG Args 1)))
             (PROG1 (LISTGET (SETQ PropList (NC.RetrievePropList Card))
                             (ARG Args 2))
                    (if (EQ Args 3)
                        then [if PropList
                                 then (LISTPUT PropList (ARG Args 2)
                                               (ARG Args 3))
                                 else (NC.SetPropList Card (LIST (ARG Args 2)
                                                                 (ARG Args 3]
                             (if (NOT (NCP.CardCachedP Card))
                                 then (NC.PutPropList Card)
                                 else (NC.SetPropListDirtyFlg Card T)))))]
            (T (NCP.ReportError 'NCP.CardProp (CONCAT Card " not an existing card or box."))
               NIL])
```

(NCP.CardAddProp

```
[LAMBDA (Card Property ItemToAdd)                                    ; Edited 3-Dec-87 19:00 by rht:
```

(* * Adds ItemToAdd to the value of ID's Property property. Returns the old value.
Same semantics as WINDOWADDPROP.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

(* * rht 5/28/87%: Fixed bug reported by Mark Frisse. Now works if PropList is NIL, as for a new card.)

```
    (if (NC.ValidCardP Card)
        then (LET* ((PropList (NC.RetrievePropList Card))
                    (OldPropValue (LISTGET PropList Property)))
                   (if (NOT (FMEMB ItemToAdd (MKLIST OldPropValue)))
                       then [if PropList
                                then (LISTPUT PropList Property (APPEND (MKLIST OldPropValue)
                                                                        (LIST ItemToAdd)))
                                else (NC.SetPropList Card `(,Property (,ItemToAdd]
                            (if (NOT (NCP.CardCachedP Card))
                                then (NC.PutPropList Card)
                                else (NC.SetPropListDirtyFlg Card T)))
                   OldPropValue)
      else (NCP.ReportError 'NCP.CardAddProp (CONCAT Card " not an existing card or box."))
           NIL])
```

(NCP.CardDelProp

```
[LAMBDA (Card Property ItemToDelete)                                 (* rht%: " 2-Mar-87 22:02")
```

(* * Deletes ItemToDelete from the Property prop of Card if it is there, returning the previous list.
If it's not there, then return NIL. Same semantics as WINDOWDELPROP.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (if (NC.ValidCardP Card)
        then (LET* ((PropList (NC.RetrievePropList Card))
                    (OldPropValue (LISTGET PropList Property)))
                   (COND
                      ((NLISTP OldPropValue)
                       NIL)
                      ((FMEMB ItemToDelete OldPropValue)
                       (LISTPUT PropList Property (for Item in OldPropValue unless (EQ Item ItemToDelete)
                                                     collect Item))
                       (if (NOT (NCP.CardCachedP Card))
                           then (NC.PutPropList Card)
                         else (NC.SetPropListDirtyFlg Card T))
                       OldPropValue)
                      (T NIL)))
      else (NCP.ReportError 'NCP.CardDelProp (CONCAT Card " not an existing card or box."))
           NIL])
```

(NCP.CardSubstance

```
[LAMBDA Args                                                         (* rht%: " 2-Mar-87 22:03")
```

(* * Return the substance for this card.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

(* * rht 8/8/86%: Now can accept one or two args. Always returns old substance, but will replace substance with second arg if present.)

(* * 8/25/86%: Now passes non-nil QuietFlg to NCP.CloseCards.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (LET (Card)
```

```
        (COND
           ((AND (NEQ Args 1)
                  (NEQ Args 2))
            (NCP.ReportError 'NCP.CardSubstance (CONCAT "Improper number of args: " Args))
            NIL)
           [(NCP.ValidCardP (SETQ Card (ARG Args 1)))
            (LET (WasActive)
                 (OR (SETQ WasActive (NCP.CardCachedP Card))
                     (NCP.CacheCards Card))
                 (PROG1 (NC.FetchSubstance Card)
                     (if (EQ Args 2)
                         then (NC.SetSubstance Card (ARG Args 2))
                              (NCP.MarkCardDirty Card))
                     (OR WasActive (NCP.CloseCards Card T)))]
           (T (NCP.ReportError 'NCP.CardSubstance (CONCAT Card " not an existing card."))
              NIL])
```

(NCP.CardRegion

```
[LAMBDA Args                                                    (* rht%: "23-May-87 22:42")
```

(* * Return the substance for this card.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

(* * rht 8/25/86%: Now passes non-nil QuietFlg to NCP.UncacheCards.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

(* * rht 5/23/87%: Now takes optional second argument which should be new region for Card.)

```
   (LET (Card)
        (COND
           ((AND (NEQ Args 1)
                  (NEQ Args 2))
            (NCP.ReportError 'NCP.CardRegion (CONCAT "Improper number of args: " Args))
            NIL)
           [(NCP.ValidCardP (SETQ Card (ARG Args 1)))
            (LET (WasActive Region)
                 (OR (SETQ WasActive (NCP.CardCachedP Card))
                     (NCP.CacheCards Card))
                 (PROG1 (NC.FetchRegion Card)
                     [if (EQ Args 2)
                         then (if (REGIONP (SETQ Region (ARG Args 2)))
                                  then (NC.SetRegion Card Region)
                                       (if (NCP.CardDisplayedP Card)
                                           then (SHAPEW (NCP.CardWindow Card)
                                                        Region)
                                                (NC.PutRegion Card))
                                  else (NCP.ReportError 'NCP.CardRegion (CONCAT Region " not a valid REGION."]
                     (OR WasActive (NCP.CloseCards Card T)))]
           (T (NCP.ReportError 'NCP.CardRegion (CONCAT Card " not an existing card."))
              NIL])
```

(NCP.CardAddText

```
[LAMBDA (Card Text Loc)                                         (* rht%: " 2-Mar-87 22:03")
```

(* * Adds the Text to ID's window at the given Loc. Loc defaults to the current cursor position.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

(* * rht 8/25/86%: Now passes non-nil QuietFlg to NCP.UncacheCards.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
   (PROG (WasActiveP)
         (if (NOT (NC.ValidCardP Card))
             then (NCP.ReportError 'NCP.CardAddText (CONCAT Card " is not an existing card or filebox."))
                  (RETURN NIL))
         (if (NOT (NCP.TextBasedP (NCP.CardType Card)))
             then (NCP.ReportError 'NCP.CardAddText "Can only add text to cards with type Text.")
                  (RETURN NIL))
         (if (NOT (SETQ WasActiveP (NCP.CardCachedP Card)))
             then (NCP.CacheCards Card))
         (NCP.ChangeLoc Card Loc)
         (TEDIT.INSERT (NC.FetchSubstance Card)
                Text)
         (NC.MarkCardDirty Card)
         (if (NOT WasActiveP)
             then (NCP.UncacheCards Card T))
         (RETURN Card])
```

(NCP.ChangeLoc

```
[LAMBDA (Card Loc)                                              (* rht%: " 2-Mar-87 22:03")
```

```
        (* * Changes the location within ID's textstream to the new loc Loc.
        Loc can be the litatoms START or END, a number, or nil. The latter indicates to use the current cursor position.
        Note that we don't mark card as dirty just because its selection changed.
        Note that this leaves card active even if inactive when we were called.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (AND (NC.ValidCardP Card)
             (NCP.TextBasedP Card))
        then (if (NOT (NCP.CardCachedP Card))
                 then (NCP.CacheCards Card))
             (LET ((Stream (NC.FetchSubstance Card)))
                  [SELECTQ Loc
                       ((START 0)
                            (TEDIT.SETSEL Stream 0 0 'RIGHT))
                       (END (TEDIT.SETSEL Stream (ADD1 (TEDIT.NCHARS Stream))
                                  0
                                  'RIGHT))
                       (AND (FIXP Loc)
                            (GEQ Loc 0)
                            (TEDIT.SETSEL Stream Loc 0 'RIGHT]
                  Card)
      else (NCP.ReportError 'NCP.ChangeLoc (CONCAT Card " not an existing card or not TEdit based."))
           NIL])
```

(NCP.DeleteCards

```
  [LAMBDA (Cards)                                                    (* rht%: " 2-Mar-87 22:04")

        (* * Delete the given cards and boxes. Or just one, if Cards is atomic.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

        (* * rht 11/7/86%: Now passes flg args to NC.DeleteNoteCards to prevent confirm messages to user.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (SETQ Cards (for Card in (MKLIST Cards) unless (if (NOT (NC.ValidCardP Card))
                                                       then (NCP.ReportError 'NCP.DeleteCards (CONCAT Card " is not
                                                                                                      an existing
                                                                                                      card or box."))

                                                            T)
                   collect Card))
    (AND Cards (NC.DeleteNoteCards Cards T NIL NIL T T])
```

(NCP.FileBoxP

```
  [LAMBDA (Card)                                                     (* rht%: "18-Nov-85 01:58")

        (* * Return T if Card is a Filebox.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

    (EQ (NCP.CardType Card)
        'FileBox])
```

(NCP.AllCards

```
  [LAMBDA (NoteFile)                                                 (* rht%: "18-Nov-85 18:19")

        (* * Return a list of IDs of all cards and boxes.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)

    (NC.MapCards NoteFile (FUNCTION PROG1)
           (FUNCTION TRUE])
```

(NCP.AllBoxes

```
  [LAMBDA (NoteFile)                                                 (* rht%: "18-Nov-85 18:19")

        (* * Return a list of all existing fileboxes.)

        (* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

    (NC.MapCards NoteFile (FUNCTION PROG1)
           (FUNCTION NCP.FileBoxP])
```

(NCP.ContentsFileBox

```
  [LAMBDA (NoteFile)                                                 (* rht%: "18-Nov-85 02:13")

        (* * Return the top level contents file box.)

        (* * rht 11/17/85%: Updated to handle new card and notefile objects.)
```

```
    (fetch (NoteFile TableOfContentsCard) of NoteFile])
```

(NCP.OrphansFileBox

```
[LAMBDA (NoteFile)                                          (* rht%: "18-Nov-85 02:14")
```

(* * Return the orphans contents file box.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

```
    (fetch (NoteFile OrphansCard) of NoteFile])
```

(NCP.ToBeFiledFileBox

```
[LAMBDA (NoteFile)                                          (* rht%: "18-Nov-85 02:15")
```

(* * Return the To be filed file box.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

```
    (fetch (NoteFile ToBeFiledCard) of NoteFile])
```

(NCP.NoteFileFromCard

```
[LAMBDA (Card)                                              (* rht%: "12-Jul-86 16:22")
```

(* * Get the card's notefile.)

```
    (AND (NC.ValidCardP Card)
         (fetch (Card NoteFile) of Card])
```

(NCP.CardNoteFile

```
[LAMBDA (Card)                                              (* rht%: "12-Jul-86 16:22")
```

(* * Get the card's notefile.)

```
    (AND (NC.ValidCardP Card)
         (fetch (Card NoteFile) of Card])
```

(NCP.SameCardP

```
[LAMBDA (Card1 Card2)                                       (* rht%: " 2-Mar-87 22:10")
```

(* * Returns non-nil if two cards are the same.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (if (AND (NCP.ValidCardP Card1)
             (NCP.ValidCardP Card2))
        then (NC.SameCardP Card1 Card2)
      else (NCP.ReportError 'NCP.SameCardP (CONCAT "Args not valid cards: " Card1 Card2])
```

(NCP.CoerceToCard

```
[LAMBDA (CardIdentifier)                                    (* rht%: " 8-Aug-86 13:50")
```

(* * Return the card object associated with CardIdentifier which can currently be any of a card object, window or text stream.)

```
    (NC.CoerceToCard CardIdentifier])
```

(NCP.DetermineDisplayRegion

```
[LAMBDA (Card Region/Position)                               (* rht%: " 8-Aug-86 12:09")
```

(* * Returns the region that Card would occupy if displayed. If Region/Position not passed in, then checks previous size or default heights and widths for the card type. Also knows about BringUpCardAtPreviousPos.)

```
    (NC.DetermineDisplayRegion Card Region/Position])
```

(NCP.LockListOfCards

```
[LAMBDA (CardIdentifiers Operation)                         (* Randy.Gobbel "23-Apr-87 14:57")
```

(* * rg |4/23/87| created)

```
    (if (NOT (BOUNDP 'CardListResetVar))
        then (NC.ReportError 'NCP.LockListOfCards "NCP.LockListOfCards called without RESETLST wrapper"))
    (NC.LockListOfCards CardIdentifiers Operation])
```

(NCP.GetCrossFileLinkDestCard

```
[LAMBDA (CrossFileLinkCard InterestedWindow Don'tOpenDestNoteFileFlg)
                                                            (* rht%: " 8-Jun-87 23:24")
```

(* * Try to get the remote card corresponding to the given cross-file link card.)

```
    (AND (NCP.ValidCardP CrossFileLinkCard)
         (EQ (NCP.CardType CrossFileLinkCard)
             'CrossFileLink)
         (NC.GetCrossFileLinkDestCard CrossFileLinkCard InterestedWindow Don'tOpenDestNoteFileFlg])
```

(NCP.CardBeingDeletedP

```
  [LAMBDA (Card)                                                    (* pmi%: "10-Dec-87 11:40")

          (* * pmi 12/10/87%: Created to provide NC.FetchBeingDeletedFlg for the programmer's interface.)

    (NC.FetchBeingDeletedFlg Card])

)

;;; Collecting, copying, moving, deleting, cards

(DEFINEQ
```

(NCP.CollectCards

```
  [LAMBDA (RootCards LinkTypes MaxDepth FollowCrossFileLinksFlg) ; Edited 11-Feb-88 11:31 by pmi
    ;; Starting from RootCards and following link of types in LinkTypes to a max depth of MaxDepth, collect and return all cards encountered.
    ;; LinkTypes can contain backward links.
    ;; rht 8/29/86: Now handles case of NULL MaxDepth.  Also handles case when RootCards is single card instead of list.
    ;; rht 6/6/87: Added FollowCrossFileLinksFlg;  if non-nil, then try to follow cross-file links into remote notefiles, else ignore them.
    ;; pmi 12/4/87: Now returns list of cards collected with NO duplicates.
    ;; rar 2/10/88: Added check for NIL RealFringe when searching for neighbors down the structure.
    ;; pmi 2/11/87: Added change in previous comment (courtesy of IDE folks).
    (OR MaxDepth (SETQ MaxDepth 65535))
    (if (LEQ MaxDepth 0)
        then RootCards
      else (LET (RealFringe Collection)
                (RESETLST
                    [RESETSAVE NIL '(PROGN (for Card in Collection do (NC.SetUserDataProp Card 'SeenBefore NIL]
                    (SETQ RealFringe (MKLIST RootCards))
                    (SETQ Collection (APPEND (MKLIST RootCards)))
                    (for Card in Collection do (NC.SetUserDataProp Card 'SeenBefore T))
                    (for Depth from 1 to MaxDepth eachtime (BLOCK) bind Fringe
                       do (SETQ Fringe (if RealFringe
                                           then (NCP.CardNeighbors RealFringe LinkTypes FollowCrossFileLinksFlg)
                                         else NIL))
                          (if (NULL Fringe)
                              then (RETURN Collection)
                            else (SETQ RealFringe (for Card in Fringe when (NOT (NC.FetchUserDataProp Card
                                                                                   'SeenBefore))
                                                     collect (NC.SetUserDataProp Card 'SeenBefore T)
                                                             (SETQ Collection (CONS Card Collection))
                                                             Card)))
                       finally (RETURN Collection)))])
```

(NCP.CopyCards

```
  [LAMBDA (Cards DestNoteFileOrFileBox RootCards QuietFlg CopyExternalToLinksMode InterestedWindow)
                                                                    ; Edited 12-Aug-88 11:01 by pmi
    ;; Make copies of Cards and all links among them, not including external links.  RootCards should be a subset of Cards or nil.  If nil, then all cards
    ;; in Cards are considered roots.  If DestNoteFileOrFileBox is a filebox, then all roots are filed in that box, otherwise, the contents box of the notefile
    ;; is used.
    ;; Note that all cards currently must be in the same notefile.
    ;; pmi 10/29/87: Added CopyExternalToLinksMode argument to be passed down to NC.CopyCards.
    ;; pmi 8/12/88: Added InterestedWindow argument.
    (NC.CopyCards Cards DestNoteFileOrFileBox RootCards QuietFlg InterestedWindow CopyExternalToLinksMode])
```

(NCP.MoveCards

```
  [LAMBDA (Cards DestNoteFileOrFileBox RootCards QuietFlg CopyExternalToLinksMode InterestedWindow)
                                                                    ; Edited 12-Aug-88 11:01 by pmi
    ;; pmi 10/29/87: Move cards into a filebox.
    ;; pmi 8/12/88: Added InterestedWindow argument.
    (NC.MoveCards Cards DestNoteFileOrFileBox RootCards QuietFlg InterestedWindow CopyExternalToLinksMode])

)

;;; Creating and accessing links

(DEFINEQ
```

(NCP.CreateLink

```
[LAMBDA (Source Destination LinkType DisplayMode)                    (* rht%: " 2-Mar-87 22:11")

          (* * Create a link from Source to Destination. Source can be a card or a list of two things;
          card and location spec. Location spec can be as in NCP.ChangeLoc documentation with the added option of the atom
          GLOBAL meaning that this should be a global link.)

    (LET (SourceLoc SourceCard)
         (COND
            ((NC.ValidCardP Source)
             (NCP.LocalGlobalLink LinkType Source Destination NIL DisplayMode))
            ((OR [NOT (NC.ValidCardP (SETQ SourceCard (CAR Source]
                 (NLISTP Source)
                 (NULL (CDR Source)))
             (NCP.ReportError 'NCP.CreateLink (CONCAT "'Source' arg is not either a card or a list of card and
                                                      atom or num: " Source)))
            ((EQ (SETQ SourceLoc (CADR Source))
                 'GLOBAL)
             (NCP.GlobalGlobalLink LinkType SourceCard Destination))
            ((OR (FMEMB (CADR Source)
                        '(START END NIL))
                 (FIXP SourceLoc))
             (NCP.LocalGlobalLink LinkType SourceCard Destination SourceLoc DisplayMode))
            (T (NCP.ReportError 'NCP.CreateLink (CONCAT "'Source' arg is not either a card or a list of card and
                                                        atom or num: " Source])
```

(NCP.LocalGlobalLink

```
[LAMBDA (LinkType SourceCard DestinationCard FromLoc DisplayMode)
                                                                     ; Edited 11-May-88 22:58 by Trigg

    ;; Create a link from within the text of the SourceCard card to the DestinationCard card.

    ;; rht 4/1/85: Changed to handle old-style link display modes.

    ;; rht 11/17/85: Updated to handle new card and notefile objects.

    ;; rht 8/25/86: Now passes non-nil QuietFlg to NCP.UncacheCards.

    ;; rht 11/16/86: Changed call to NCP.ReportError

    ;; rht 5/27/87: Changed call to NCP.CoerceToLinkDisplayMode slightly.

    ;; pmi 4/8/88: Now checks if the SourceCard is graph-based, and if so, calls NC.InsertLinkInGraph, which now takes a position argument
    ;; (FromLoc).

    ;; rht 5/11/88: Was returning non-nil when NCP.FileCards returned NIL.

    (DECLARE (GLOBALVARS NC.SubBoxLinkLabel NC.FiledCardLinkLabel))
    (PROG ((CoercedDisplayMode (NCP.CoerceToLinkDisplayMode (OR DisplayMode SourceCard)))
           WasActive NoteFile)
          (if (NULL CoercedDisplayMode)
              then (NCP.ReportError 'NCP.LocalGlobalLink (CONCAT DisplayMode " is invalid link display mode."
                                                                 (CHARACTER 13)
                                                                 "No link created."))
                   (RETURN NIL))
          (if (NCP.GraphBasedP SourceCard)
              then (if [NOT (FMEMB LinkType (NCP.LinkTypes (SETQ NoteFile (fetch (Card NoteFile) of SourceCard]
                       then (if (NC.AskYesOrNo (CONCAT "That link type hasn't been used in NoteFile "
                                                       (fetch (NoteFile FullFileName) of NoteFile)
                                                       (CHARACTER 13)
                                                       "Want to create a new link type: " LinkType "? ")
                                "--" NIL T (NC.AttachPromptWindow (WFROMMENU (fetch (NoteFile Menu)
                                                                                    of NoteFile)))
                                NIL NIL)
                            then (NCP.CreateLinkType LinkType NoteFile)
                           else (RETURN NIL)))
                   (OR (SETQ WasActive (NCP.CardCachedP SourceCard))
                       (NCP.CacheCards SourceCard))
                   [RETURN (PROG1 (NC.InsertLinkInGraph SourceCard LinkType DestinationCard CoercedDisplayMode
                                                        FromLoc)
                                  (OR WasActive (NCP.UncacheCards SourceCard T)))]
             elseif (NCP.TextBasedP SourceCard)
              then (if (EQ LinkType NC.FiledCardLinkLabel)
                       then (if (AND (NOT (NCP.FileBoxP DestinationCard))
                                     (NCP.FileBoxP SourceCard))
                                then (OR (SETQ WasActive (NCP.CardCachedP SourceCard))
                                         (NCP.CacheCards SourceCard))
                                     (NCP.ChangeLoc SourceCard FromLoc)
                                     [RETURN (PROG1 (if (NCP.FileCards DestinationCard SourceCard)
                                                        then (CAR (NCP.GetLinks SourceCard DestinationCard
                                                                                NC.FiledCardLinkLabel))
                                                      else NIL)
                                                    (OR WasActive (NCP.UncacheCards SourceCard T)))]
                              else (NCP.ReportError 'NCP.LocalGlobalLink (CONCAT "FiledCard link must be from a box
                                                                                 to a card." (CHARACTER 13)
                                                                                 "No link created."))
                                   (RETURN NIL)))
                   (if (EQ LinkType NC.SubBoxLinkLabel)
                       then (if (AND (NCP.FileBoxP DestinationCard)
                                     (NCP.FileBoxP SourceCard))
                                then (OR (SETQ WasActive (NCP.CardCachedP SourceCard))
                                         (NCP.CacheCards SourceCard))
```

```
                                (NCP.ChangeLoc SourceCard FromLoc)
                                [RETURN (PROG1 (if (NCP.FileCards DestinationCard SourceCard)
                                                   then (CAR (NCP.GetLinks SourceCard DestinationCard
                                                                           NC.SubBoxLinkLabel))
                                                   else NIL)
                                               (OR WasActive (NCP.UncacheCards SourceCard T)))]
                           else (NCP.ReportError 'NCP.LocalGlobalLink (CONCAT "SubBox link must be from a box to
                                                                              a box." (CHARACTER 13)
                                                                              "No link created."))
                                (RETURN NIL)))                    ; Inserting non-hierarchical link into a filebox.
                     (if (NCP.FileBoxP SourceCard)
                         then (NCP.ReportError 'NCP.LocalGlobalLink (CONCAT "Local links from fileboxes must be
                                                                            either SubBox or FiledCard." (CHARACTER
                                                                                                          13)
                                                                            "No link created."))
                              (RETURN NIL))
                     (if [NOT (FMEMB LinkType (NCP.LinkTypes (SETQ NoteFile (fetch (Card NoteFile) of SourceCard]
                         then (if (NC.AskYesOrNo (CONCAT "That link type hasn't been used in NoteFile "
                                                         (fetch (NoteFile FullFileName) of NoteFile)
                                                         (CHARACTER 13)
                                                         "Want to create a new link type: " LinkType "? ")
                                  "--" NIL T (NC.AttachPromptWindow (WFROMMENU (fetch (NoteFile Menu)
                                                                                      of NoteFile)))
                                  NIL NIL)
                                  then (NCP.CreateLinkType LinkType NoteFile)
                                  else (RETURN NIL)))
                     (OR (SETQ WasActive (NCP.CardCachedP SourceCard))
                         (NCP.CacheCards SourceCard))
                     (AND FromLoc (NCP.ChangeLoc SourceCard FromLoc))
                     (RETURN (PROG1 (NC.InsertLinkInText (NC.FetchSubstance SourceCard)
                                                         LinkType DestinationCard SourceCard CoercedDisplayMode)
                                    (OR WasActive (NCP.UncacheCards SourceCard T)))])
```

(NCP.GlobalGlobalLink

```
[LAMBDA (LinkType FromCard ToCard)                    (* rht%: " 2-Mar-87 22:04")
```

(* * This builds a global link of type LinkType between FromCard and ToID.
Complains if link type is system-defined with restricted semantics.
If LinkType is brand new, then asks if user wants to create a new label by that name.)

(* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

(* * kirk 23Jan86 Changed to use NC.AskYesOrNo)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
  (LET ((NoteFile (fetch (Card NoteFile) of FromCard))
        LinkTypes)
       (SETQ LinkTypes (NCP.UserLinkTypes NoteFile))
       (COND
          ((OR (EQ LinkType NC.SourceLinkLabel)
               (FMEMB LinkType LinkTypes))
```

(* * LinkType type is okay so create the link.)

```
           (NC.MakeGlobalLink NIL LinkType ToCard FromCard))
          ((FMEMB (U-CASE LinkType)
                  NC.UCASESystemLinkLabels)
           (NCP.ReportError 'NCP.GlobalGlobalLink (CONCAT "Can't make a global-to-global link of type " LinkType
                                                          "."))
           NIL)
          ((NC.AskYesOrNo (CONCAT "That link type hasn't been used in this NoteFile." (CHARACTER 13)
                                  "Want to create a new link type: " LinkType "? ")
                  "--" NIL T (NC.AttachPromptWindow (WFROMMENU (fetch (NoteFile Menu) of NoteFile)))
                  NIL NIL)
           (NCP.CreateLinkType LinkType NoteFile)
```

(* * LinkType type is okay so create the link.)

```
           (NC.MakeGlobalLink NIL LinkType ToCard FromCard))
          (T NIL])
```

(NCP.GlobalLocalLink

```
[LAMBDA (LinkType FromCard ToCard ToLoc)              (* rht%: " 2-Mar-87 22:11")
```

(* * Builds a link from the FromID card as a whole to within the text of the ToID card.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
  (NCP.ReportError 'NCP.GlobalLocalLink "Sorry, can't make global to local links yet."])
```

(NCP.LocalLocalLink

```
[LAMBDA (LinkType FromCard ToCard FromLoc ToLoc)      (* rht%: " 2-Mar-87 22:11")
```

(* * Builds a link from within the text of the FromID card to within the text of the ToID card.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
(NCP.ReportError 'NCP.LocalLocalLink "Sorry, can't make local to local links yet."])
```

(NCP.LinkDesc

```
[LAMBDA (Link FollowCrossFileLinkFlg)                    (* rht%: "29-Oct-87 18:59")
```

(* * Return a list structure which describes Link. It consists of (label <fromdesc> <todesc>) where <fromdesc> and <todesc> describe the anchoring of the link at each end. They each have the form%:
(<anchormode> <ID> <loc>))

(* * rht 11/16/86%: Changed call to NCP.ReportError)

(* * rht 10/29/87%: Now takes FollowCrossFileLinkFlg arg.)

```
(COND
   ((NCP.ValidLink Link)
    (LIST (fetch (Link Label) of Link)
          (NCP.LinkAnchorDesc Link NIL FollowCrossFileLinkFlg)
          (NCP.LinkAnchorDesc Link T FollowCrossFileLinkFlg)))
   (T (NCP.ReportError 'NCP.LinkDesc (CONCAT "No such link: " Link])
```

(NCP.LinkDisplayMode

```
[LAMBDA Args                                             (* rht%: " 2-Mar-87 22:05")
```

(* * Takes either 1 or 2 args. The first is a link, the second an optional new link display mode.
Return old display mode in any case; change mode if the second arg is present.)

(* * rht 7/12/86%: Now takes a list of three elements for new displaymode rather than an instance of the LINKDISPLAYMODE record. Also calls NCP.CoerceToLinkDisplayMode.)

(* * rht 8/25/86%: Now passes non-nil QuietFlg to NCP.UncacheCards.)

```
(LET (Link NewMode WasActiveFlg SourceCard)
     (COND
        ((AND (NEQ Args 1)
              (NEQ Args 2))
         (NCP.ReportError 'NCP.LinkDisplayMode (CONCAT "Improper number of args: " Args))
         NIL)
        [(NCP.ValidLink (SETQ Link (ARG Args 1)))
         (PROG1 (fetch (Link DisplayMode) of Link)
             [if (EQ Args 2)
                 then (if (SETQ NewMode (NCP.CoerceToLinkDisplayMode (ARG Args 2)))
                          then (OR [SETQ WasActiveFlg (NCP.CardCachedP (SETQ SourceCard (fetch (Link
                                                                                                SourceCard
                                                                                                )
                                                                                       of Link]
                                                       (NCP.CacheCards SourceCard))
                                   (NC.ChangeLinkDisplayMode Link NIL NewMode)
                                   (OR WasActiveFlg (NCP.UncacheCards SourceCard T))
                      else (NCP.ReportError 'NCP.LinkDisplayMode (CONCAT (ARG Args 2)
                                                                         " is invalid link display mode."])]
        (T (NCP.ReportError 'NCP.LinkDisplayMode (CONCAT Link " is not a valid link."))
           NIL])
```

(NCP.LinkType

```
[LAMBDA Args                                             (* rht%: " 2-Mar-87 22:05")
```

(* * Takes either 1 or 2 args. The first is a link, the second an optional new label.
Return old label in any case; change label if the second arg is present.)

(* * rht 2/8/85%: Now makes sure source card of link is active before calling NC.RelabelLink.)

(* * rht 8/25/86%: Now passes non-nil QuietFlg to NCP.UncacheCards.)

```
(LET (Link NewLinkType SourceCard NoteFile)
     (COND
        ((AND (NEQ Args 1)
              (NEQ Args 2))
         (NCP.ReportError 'NCP.LinkType (CONCAT "Improper number of args: " Args))
         NIL)
        [(NCP.ValidLink (SETQ Link (ARG Args 1)))
         (PROG1 (fetch (Link Label)
                       Link)
             [if (EQ Args 2)
                 then (COND
                         ((FMEMB (SETQ NewLinkType (ARG Args 2))
                                 NC.SystemLinkLabels)
                          (NCP.ReportError 'NCP.LinkType (CONCAT "Can't change label to a system label: "
                                                                 NewLinkType)))
                         ((OR [FMEMB NewLinkType (NCP.LinkTypes (SETQ NoteFile (fetch (Card NoteFile)
                                                                                   of (SETQ SourceCard
```

```
                                                                                    (fetch (Link SourceCard)
                                                                                      of Link]
                                        (AND (NC.AskYesOrNo (CONCAT "That link type hasn't been used in this
                                                                    NoteFile." (CHARACTER 13)
                                                                    "Want to create a new link type: " NewLinkType
                                                                    "? ")
                                                    "--" NIL T NIL NIL NIL)
                                             (NCP.CreateLinkType NewLinkType NoteFile)))
                                   (if (NCP.CardCachedP SourceCard)
                                       then (NC.RelabelLink Link NIL NewLinkType T)
                                     else (NCP.CacheCards SourceCard)
                                          (NC.RelabelLink Link NIL NewLinkType T)
                                          (NCP.UncacheCards SourceCard T])]
              (T (NCP.ReportError 'NCP.LinkType (CONCAT Link " is not a valid link."))
                 NIL])
```

(NCP.LinkSource

```
[LAMBDA (Link)                                                    (* rht%: " 2-Mar-87 22:11")

        (* * Return the SOURCEID of Link.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

   (if (NCP.ValidLink Link)
       then (fetch (Link SourceCard) of Link)
     else (NCP.ReportError 'NCP.LinkSource (CONCAT Link " not an existing link."])
```

(NCP.LinkDestination

```
[LAMBDA (Link)                                                    (* rht%: " 2-Mar-87 22:11")

        (* * Return the DESTINATIONID of Link.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

   (COND
      ((NCP.ValidLink Link)
       (fetch (Link DestinationCard) of Link))
      (T (NCP.ReportError 'NCP.LinkDestination (CONCAT Link " not an existing link."])
```

(NCP.DeleteLinks

```
[LAMBDA (Links)                                                   (* rht%: " 2-Mar-87 22:11")

        (* * Delete each link in Links. If Links is one link, then just delete that one.)

        (* * rht 11/18/85%: Updated to call new NC.DeleteLink function.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

   (for Link in (COND
                   ((type? Link Links)
                    (LIST Links))
                   (T Links))
      do (if (NCP.ValidLink Link)
             then (NC.DeleteLink Link)
           else (NCP.ReportError 'NCP.DeleteLinks (CONCAT "No such link: " Link])
```

(NCP.ValidLinkP

```
[LAMBDA (Link)                                                    (* rht%: "18-Jul-86 16:49")

        (* * True if Link is an extant link in current database.)

   (NC.ValidLinkP Link])
```

(NCP.AllLinks

```
[LAMBDA (NoteFile)                                                (* rht%: "18-Feb-87 12:00")

        (* * Return a list of all links in the current database.)

        (* * rht 11/18/85%: Updated to handle new notefile and card object formats.
        Note that it doesn't call NCONC to flatten the lists in case someone else is holding a pointer to the value returned by
        NC.RetrieveToLinks.)

        (* * rht 2/18/87%: Changed to call MAPCONC instead of APPLY'ing APPEND.
        Note call to APPEND to prevent flatten'ing of original lists.)

   (MAPCONC (NC.MapCards NoteFile (FUNCTION NC.RetrieveToLinks)
                  (FUNCTION TRUE))
          (FUNCTION (LAMBDA (X)
                      (APPEND X])
```

(NCP.SameLinkP

```
  [LAMBDA (Link1 Link2)                                              (* rht%: " 2-Mar-87 22:12")

          (* * Returns non-nil if two links are the same.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (AND (NCP.ValidLinkP Link1)
             (NCP.ValidLinkP Link2))
        then (NC.SameCardP Link1 Link2)
      else (NCP.ReportError 'NCP.SameLinkP (CONCAT "Args not valid links: " Link1 Link2])
```

(NCP.LinkFromLinkIcon

```
  [LAMBDA (LinkIcon)                                                 (* rht%: " 8-Aug-86 12:14")

          (* * If LinkIcon is an image object for a link icon, return the associated link.)

    (if (NC.LinkIconImageObjP LinkIcon)
        then (IMAGEOBJPROP LinkIcon 'OBJECTDATUM])
```

(NCP.MakeLinkIcon

```
  [LAMBDA (Link)                                                     (* rht%: " 8-Aug-86 12:18")

          (* * Return a link icon image object for Link.)

    (NC.MakeLinkIcon Link])
```

(NCP.MarkCardDirty

```
  [LAMBDA (Card ResetFlg)                                            (* rht%: "10-Sep-86 15:39")

          (* * Mark Card's substance as dirty thus forcing it to be written down at the next save.
          If ResetFlg is non-nil, then unmark it as dirty.)

          (* * rht 9/10/86%: Fixed typo. Card wasn't being passed to NC.MarkCardDirty.)

    (NC.MarkCardDirty Card ResetFlg])
```

(NCP.LinkIconAttachedBitMap

```
  [LAMBDA (TypeName Size)                                            (* pmi%: "31-Dec-87 13:32")

          (* * dsj. This returns the link iconattached bitmap of size Size for card type Type.
          Default size is 17 x 17.0)

          (* * pmi 12/30/87%: Added dsj's function to the programmer's interface;
          see comment above. Placed default value in new globalvar NCP.DefaultLinkIconAttachedBitMapSize.)

    (DECLARE (GLOBALVARS NCP.DefaultLinkIconAttachedBitMapSize))
    (LET [(BitMap (NCP.CardTypeVar TypeName 'LinkIconAttachedBitMap]
         (AND BitMap (OR (BITMAPP BitMap)
                         (LISTGET BitMap (OR Size (NUMBERP NCP.DefaultLinkIconAttachedBitMapSize])

)


;;; Creating and accessing link labels.

(DEFINEQ
```

(NCP.CreateLinkType

```
  [LAMBDA (LinkType NoteFile QuietFlg)                               (* pmi%: "13-Jan-88 15:07")

          (* * Create a new link label unless already defined.)

          (* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

          (* * rht 11/16/86%: Changed call to NCP.ReportError)

          (* * pmi 1/13/88%: Added QuietFlg argument to control printing of error message)

    (if (FMEMB LinkType (NCP.LinkTypes NoteFile))
        then (OR QuietFlg (NCP.ReportError 'NCP.CreateLinkType (CONCAT "Link type already defined: " LinkType "."))
                 ))
             NIL
      else (NC.StoreLinkLabels NoteFile (CONS LinkType (NCP.UserLinkTypes NoteFile)))
           LinkType])
```

(NCP.DeleteLinkType

```
  [LAMBDA (LinkType NoteFile)                                        (* rht%: " 2-Mar-87 22:06")

          (* * Checks for any instance of LinkType in the current database.
          If can't find any then delete the link label, otherwise error out.)

          (* * rht 11/18/85%: Updated to handle new notefile and card object formats.)
```

(* * rht 11/16/86%: Changed call to NCP.ReportError)

(* * rht 2/18/87%: Fixed call to NCP.AllLinks to take NoteFile arg.)

```
(DECLARE (GLOBALVARS NC.SystemLinkLabels))
(COND
   ((NOT (FMEMB LinkType (NCP.LinkTypes NoteFile)))
    (NCP.ReportError 'NCP.DeleteLinkType (CONCAT "No such link type: " LinkType "."))
    NIL)
   ((FMEMB LinkType NC.SystemLinkLabels)
    (NCP.ReportError 'NCP.DeleteLinkType (CONCAT "Can't delete system link type: " LinkType "."))
    NIL)
   ((for Link in (NCP.AllLinks NoteFile) thereis (EQ LinkType (fetch (Link Label) of Link)))
    (NCP.ReportError 'NCP.DeleteLinkType (CONCAT "Link type currently in use: " LinkType ". Can't delete."))
    NIL)
   (T (NC.StoreLinkLabels NoteFile (REMOVE LinkType (NCP.UserLinkTypes NoteFile)))
      LinkType])
```

(NCP.RenameLinkType

```
[LAMBDA (OldLinkType NewLinkType NoteFile)                    (* rht%: " 2-Mar-87 22:07")
```

(* * Renames all instances of links with OldLinkType to be NewLabel.
And deletes the old label OldLabel. If NewLinkType doesn't exist, create it.)

(* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

(* * rht 11/11/86%: Took out ENV args to FUNCTION because they seemed to be causing stack overflows.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
   (LET ((LinkTypes (NCP.LinkTypes NoteFile)))
        (COND
           ((NOT (FMEMB OldLinkType LinkTypes))
            (NCP.ReportError 'NCP.RenameLinkType (CONCAT "No such link type: " OldLinkType "."))
            NIL)
           ((FMEMB OldLinkType NC.SystemLinkLabels)
            (NCP.ReportError 'NCP.RenameLinkType (CONCAT "Can't rename system link type: " OldLinkType "."))
            NIL)
           ((FMEMB NewLinkType NC.SystemLinkLabels)
            (NCP.ReportError 'NCP.RenameLinkType (CONCAT "Can't rename with a system link type: " NewLinkType
                                                         "."))
            NIL)
           (T (COND
                 ((NOT (FMEMB NewLinkType LinkTypes))
                  (NCP.CreateLinkType NewLinkType NoteFile)))     (* Map down all links, relabeling as appropriate.)
              [NCP.MapLinks NoteFile (FUNCTION (LAMBDA (Link)
                                               (COND
                                                  ((EQ OldLinkType (fetch (Link Label) of Link))
                                                   (NCP.LinkType Link NewLinkType]
              (NC.StoreLinkLabels NoteFile (REMOVE OldLinkType (NCP.UserLinkTypes NoteFile)))
              NewLinkType])
```

(NCP.LinkTypes

```
[LAMBDA (NoteFile)                                            (* rht%: " 9-Dec-85 15:37")
```

(* * Return all link labels including system ones.)

(* * rht 11/17/85%: Updated to handle new card and notefile objects.)

```
   (NC.RetrieveLinkLabels NoteFile T])
```

(NCP.ReverseLinkTypes

```
[LAMBDA (NoteFile)                                            (* rht%: "18-Nov-85 17:33")
```

(* * Return all reverse link labels including system ones.)

(* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

```
   (for Lab in (NC.RetrieveLinkLabels NoteFile T) collect (PACK* '_ Lab])
```

(NCP.UserLinkTypes

```
[LAMBDA (NoteFile)                                            (* rht%: "18-Nov-85 17:34")
```

(* * Return list of only the user defined link labels appearing in the current notefile.)

(* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

```
   (NC.RetrieveLinkLabels NoteFile NIL])
```

(NCP.ValidLinkTypeP

```
[LAMBDA (LinkType NoteFile)                                   (* rht%: "12-Jul-86 12:32")
```

```
        (* * True if LinkType is a currently defined user or system link label.)

        (* * rht 11/18/85%: Updated to handle new notefile and card object formats.)

    (FMEMB LinkType (NCP.LinkTypes NoteFile])
```

(NCP.SystemLinkTypeP

```
  [LAMBDA (LinkType)                                    (* pmi%: " 8-Dec-87 15:36")

        (* * dsj 10/24/87%: Is LinkType a system link label?)

        (* * pmi 12/8/87%: Added dsj's function; see comment above.)

    (NC.SystemLinkLabelP LinkType])

)

;;; Dealing with card parts dates.

(DEFINEQ
```

(NCP.CardDates

```
  [LAMBDA (Card)                                        (* rht%: " 2-Mar-87 22:07")

        (* * Returns an instance of the NOTECARDDATES record filled in with the current dates of the card parts of Card.)

        (* * rht 8/25/86%: Now passes non-nil QuietFlg to NCP.UncacheCards.)

        (* * rht 11/16/86%: Changed call to NCP.ReportError)

    (if (NC.ValidCardP Card)
        then [LET ((WasActive (NCP.CardCachedP Card)))
                  (OR WasActive (NCP.CacheCards Card))
                  (PROG1 (create NOTECARDDATES
                                 SUBSTANCEDATE _ (NC.FetchItemDate Card)
                                 LINKSDATE _ (NC.FetchLinksDate Card)
                                 TITLEDATE _ (NC.FetchTitleDate Card)
                                 PROPLISTDATE _ (NC.FetchPropListDate Card))
                         (OR WasActive (NCP.UncacheCards Card T)))]
      else (NCP.ReportError 'NCP.CardDates (CONCAT Card " not an existing card."))
           NIL])

)

;;; Open events card

(DEFINEQ
```

(NCP.GetOpenEventsCard

```
  [LAMBDA (NoteFile)                                    (* rht%: " 8-Dec-86 21:31")

        (* * Find the open events card for the given notefile, creating a new one if necessary.
        Mark the card as not needing to be filed.)

    (LET ((OpenEventsCard (NCP.LookupCardByName 'OpenEventsCard NoteFile)))
         (if (NOT (NCP.ValidCardP OpenEventsCard))
             then (SETQ OpenEventsCard (NCP.CreateCard 'List NoteFile "Open Events" T))
                  (NCP.MarkAsNotNeedingFiling OpenEventsCard)
                  (NCP.RegisterCardByName 'OpenEventsCard OpenEventsCard))
         OpenEventsCard])
```

(NCP.GetCloseEventsCard

```
  [LAMBDA (NoteFile)                                    (* rht%: " 8-Dec-86 21:31")

        (* * Find the close events card for the given notefile, creating a new one if necessary.
        Mark the card as not needing to be filed.)

    (LET ((CloseEventsCard (NCP.LookupCardByName 'CloseEventsCard NoteFile)))
         (if (NOT (NCP.ValidCardP CloseEventsCard))
             then (SETQ CloseEventsCard (NCP.CreateCard 'List NoteFile "Close Events" T))
                  (NCP.MarkAsNotNeedingFiling CloseEventsCard)
                  (NCP.RegisterCardByName 'CloseEventsCard CloseEventsCard))
         CloseEventsCard])
```

(NCP.MarkAsNotNeedingFiling

```
  [LAMBDA (Card)                                        (* rht%: " 8-Dec-86 21:34")

        (* * Indicate on the card's prop list that it doesn't have to be filed.)

    (NCP.CardProp Card 'Don'tRequireFilingFlg T])
```

```
)
```

;;; Functions for adding menu items

```
(DEFINEQ
```

(NCP.AddSessionIconMenuItem

```
  [LAMBDA (MenuName Item)                                    (* pmi%: "13-Jul-87 17:51")
```

(* * pmi 7/13/87%: New function which allows users to add items to the Session Icon menus.
MenuName should be either Card, NoteFile, or Other. Item should be a full menu item.)

```
    (DECLARE (GLOBALVARS NC.CardOpsItems NC.CardOpsMenu NC.NoteFileOpsItems NC.NoteFileOpsMenu NC.OtherOpsItems
                    NC.OtherOpsMenu))
```

(* * Make sure we have a full menu item.)

(* * Make sure MenuName in non-NIL)

```
    (if (NOT (LISTP Item))
        then (NC.ReportError 'NCP.AddSessionIconMenuItem "You must provide a FULL menu item.")
             NIL
      elseif (NULL MenuName)
        then NIL
      else (SELECTQ MenuName
                (Card (if (NOT (MEMBER Item NC.CardOpsItems))
                          then (SETQ NC.CardOpsItems (APPEND NC.CardOpsItems (LIST Item)))
                               (SETQ NC.CardOpsMenu NIL)))
                (NoteFile
```

(* * Since the NoteFile menu is different from the other two, we have to play games to set it up correctly.)

```
                          (if (for MenuItem in NC.NoteFileOpsItems do (if (EQ (CAR Item)
                                                                              (CAR MenuItem))
                                                                          then (RETURN NIL))
                                  finally (RETURN T))
                              then                                  (* This item has not already been added to this menu, so add it.)
                                   [SETQ NC.NoteFileOpsItems (APPEND NC.NoteFileOpsItems (LIST (
                                                                     NC.CreateSessionIconNoteFileMenuItem
                                                                                                  Item]
                                   (SETQ NC.NoteFileOpsMenu NIL)))
                (Other (if (NOT (MEMBER Item NC.OtherOpsItems))
                           then (SETQ NC.OtherOpsItems (APPEND NC.OtherOpsItems (LIST Item)))
                                (SETQ NC.OtherOpsMenu NIL)))
                NIL)
          Item])
```

(NCP.RemoveSessionIconMenuItem

```
  [LAMBDA (MenuName ItemName)                                (* pmi%: "13-Jul-87 18:11")
```

(* * pmi 7/13/87%: New function which allows users to remove an item from the Session Icon menus.
MenuName should be either Card, NoteFile, or Other. ItemName should be just the name of the item
(the first part of a full menu item.))

```
    (DECLARE (GLOBALVARS NC.CardOpsItems NC.CardOpsMenu NC.NoteFileOpsItems NC.NoteFileOpsMenu NC.OtherOpsItems
                    NC.OtherOpsMenu))
```

(* * Make sure we have only the name of a menu item.)

(* * Make sure MenuName in non-NIL)

```
    (if (LISTP ItemName)
        then (NC.ReportError 'NCP.RemoveSessionIconMenuItem "You must provide only the name of the menu item as
                    an atom.")
             NIL
      elseif (NULL MenuName)
        then NIL
      else
```

(* * Select the appropriate menu.)

```
          (SELECTQ MenuName
                (Card (for MenuItem in NC.CardOpsItems when (AND (LISTP MenuItem)
                                                                 (EQ (CAR MenuItem)
                                                                     ItemName))
                          do (SETQ NC.CardOpsItems (REMOVE MenuItem NC.CardOpsItems))
                             (SETQ NC.CardOpsMenu NIL)
                             (RETURN MenuItem)))
                (NoteFile (for MenuItem in NC.NoteFileOpsItems when (AND (LISTP MenuItem)
                                                                         (EQ (CAR MenuItem)
                                                                             ItemName))
                             do (SETQ NC.NoteFileOpsItems (REMOVE MenuItem NC.NoteFileOpsItems))
                                (SETQ NC.NoteFileOpsMenu NIL)
                                (RETURN MenuItem)))
                (Other (for MenuItem in NC.OtherOpsItems when (AND (LISTP MenuItem)
```

```
                                                        (EQ (CAR MenuItem)
                                                            ItemName))
                        do (SETQ NC.OtherOpsItems (REMOVE MenuItem NC.OtherOpsItems))
                           (SETQ NC.OtherOpsMenu NIL)
                           (RETURN MenuItem)))
            NIL])
```

(NCP.RestoreSessionIconMenu

```
[LAMBDA (MenuName)                                          (* pmi%: "13-Jul-87 18:12")
```

(* * pmi 7/13/87%: New function which allows users to restore the original menus of the Session Icon.
MenuName should be either Card, NoteFile, Other or NIL. NIL Signifies to restore all three of the Session Icon menus.)

```
  (DECLARE (GLOBALVARS NC.InitialCardOpsItems NC.CardOpsItems NC.CardOpsMenu NC.InitialNoteFileOpsItems
                  NC.NoteFileOpsItems NC.NoteFileOpsMenu NC.InitialOtherOpsItems NC.OtherOpsItems
                  NC.OtherOpsMenu))
  (SELECTQ MenuName
      (Card (SETQ NC.CardOpsItems NC.InitialCardOpsItems)
            (SETQ NC.CardOpsMenu NIL))
      (NoteFile (SETQ NC.NoteFileOpsItems NC.InitialNoteFileOpsItems)
                (SETQ NC.NoteFileOpsMenu NIL))
      (Other (SETQ NC.OtherOpsItems NC.InitialOtherOpsItems)
             (SETQ NC.OtherOpsMenu NIL))
      (NIL (SETQ NC.CardOpsItems NC.InitialCardOpsItems)
           (SETQ NC.CardOpsMenu NIL)
           (SETQ NC.NoteFileOpsItems NC.InitialNoteFileOpsItems)
           (SETQ NC.NoteFileOpsMenu NIL)
           (SETQ NC.OtherOpsItems NC.InitialOtherOpsItems)
           (SETQ NC.OtherOpsMenu NIL))
      NIL])
```

(NCP.AddNoteFileIconMenuItem

```
[LAMBDA (Item OpenOrClosedOrBoth)                           ; Edited  3-Dec-87 19:01 by rht:
```

(* * pmi 7/14/87%: New function which allows users to add items to the NoteFile Icon menu.
Item should be a full 3-part menu item. OpenOrClosedOrBoth should be one of the atoms%: Open, Closed
(\, or) Both, to indicate whether the new item applies to open or closed notefiles.)

```
  (DECLARE (GLOBALVARS NC.NoteFileIconOperationsMenuItems NC.NoteFileIconOpenOperations
                  NC.NoteFileIconCloseOperations))
  (PROG (ItemName)
```

(* * Make sure OpenOrClosedOrBoth in non-NIL)

```
        (if (NULL OpenOrClosedOrBoth)
            then (RETURN NIL))
```

(* * Make sure we have a full menu item.)

```
        (if (NOT (LISTP Item))
            then (NC.ReportError 'NCP.AddNoteFileIconMenuItem "You must provide a FULL menu item.")
                 (RETURN NIL))
        (SETQ ItemName (CAR Item))
```

(* * If this item is not already known, add it to the list.)

```
        [if (NOT (MEMBER Item NC.NoteFileIconOperationsMenuItems))
            then (SETQ NC.NoteFileIconOperationsMenuItems (APPEND NC.NoteFileIconOperationsMenuItems
                                                                  (LIST Item]
```

(* * Add the item as either for open notefiles, closed notefiles, or both.)

```
        (SELECTQ OpenOrClosedOrBoth
            (Open                                           (* Add the item to the open notefile list, if it isn't already there.)
                  (if (NOT (MEMBER ItemName NC.NoteFileIconOpenOperations))
                      then (SETQ NC.NoteFileIconOpenOperations (CONS ItemName NC.NoteFileIconOpenOperations)))
                                                            (* Remove the item from the closed notefile list.)
                  (SETQ NC.NoteFileIconCloseOperations (REMOVE ItemName NC.NoteFileIconCloseOperations)))
            (Closed                                         (* Add the item to the closed notefile list, if it isn't already there.)
                    (if (NOT (MEMBER ItemName NC.NoteFileIconCloseOperations))
                        then (SETQ NC.NoteFileIconCloseOperations (CONS ItemName
                                                                        NC.NoteFileIconCloseOperations)))
                                                            (* Remove the item from the open notefile list.)
                    (SETQ NC.NoteFileIconOpenOperations (REMOVE ItemName NC.NoteFileIconOpenOperations)))
            (Both                                           (* Add the item to the open notefile list, if it isn't already there.)
                  (if (NOT (MEMBER ItemName NC.NoteFileIconOpenOperations))
                      then (SETQ NC.NoteFileIconOpenOperations (CONS ItemName NC.NoteFileIconOpenOperations)))
                                                            (* Add the item to the closed notefile list, if it isn't already there.)
                  (if (NOT (MEMBER ItemName NC.NoteFileIconCloseOperations))
                      then (SETQ NC.NoteFileIconCloseOperations (CONS ItemName NC.NoteFileIconCloseOperations)
                                 )))
            (RETURN NIL))
```

(* * Recompute the menus.)

```
          (NC.MakeNoteFileIconOperationsMenus)
          (RETURN Item])
```

(NCP.RemoveNoteFileIconMenuItem

```
[LAMBDA (ItemName)                                          (* pmi%: "14-Jul-87 11:23")
```

(* * pmi 7/14/87%: New function which allows users to remove an item from the NoteFile Icon menu.
ItemName should be either the full menu item or just the name of the item
(the first part of a full menu item.))

```
    (DECLARE (GLOBALVARS NC.NoteFileIconOperationsMenuItems NC.NoteFileIconOpenOperations
                    NC.NoteFileIconCloseOperations))
```

(* * Make sure we have only the name of a menu item.)

```
    (if (LISTP ItemName)
        then (NC.ReportError 'NCP.AddSessionIconMenuItem "You must provide only the name of the menu item as an
                    atom.")
             NIL
      else
```

(* * (NIL Find) the full menu item and remove it.)

```
          (for MenuItem in NC.NoteFileIconOperationsMenuItems when (AND (LISTP MenuItem)
                                                                        (EQ (CAR MenuItem)
                                                                            ItemName))
             do (SETQ NC.NoteFileIconOperationsMenuItems (REMOVE MenuItem NC.NoteFileIconOperationsMenuItems))
                (SETQ NC.NoteFileIconOpenOperations (REMOVE ItemName NC.NoteFileIconOpenOperations))
                (SETQ NC.NoteFileIconCloseOperations (REMOVE ItemName NC.NoteFileIconCloseOperations))
                (NC.MakeNoteFileIconOperationsMenus)
                (RETURN MenuItem])
```

(NCP.RestoreNoteFileIconMenu

```
[LAMBDA NIL                                                 (* pmi%: "13-Jul-87 20:59")
```

(* * pmi 7/13/87%: New function which allows users to restore the NoteFile Icon menu to its original state.)

```
    (DECLARE (GLOBALVARS NC.InitialNoteFileIconOperationsMenuItems NC.NoteFileIconOperationsMenuItems
                    NC.InitialNoteFileIconOpenOperations NC.NoteFileIconOpenOperations
                    NC.InitialNoteFileIconCloseOperations NC.NoteFileIconCloseOperations))
    (SETQ NC.NoteFileIconOperationsMenuItems NC.InitialNoteFileIconOperationsMenuItems)
    (SETQ NC.NoteFileIconOpenOperations NC.InitialNoteFileIconOpenOperations)
    (SETQ NC.NoteFileIconCloseOperations NC.InitialNoteFileIconCloseOperations)
    (NC.MakeNoteFileIconOperationsMenus])
```

(NC.CreateSessionIconNoteFileMenuItem

```
[LAMBDA (Item)                                              ; Edited  3-Dec-87 19:01 by rht:
```

(* * pmi 7/10/87%: Created to format the menu item for the NoteFile menu of the Session Icon.)

```
    (LET (MenuItem SubItemList SubItems)
         [SETQ MenuItem `(,(CAR Item)
                          [NC.DoNoteFileOp ',(CADR Item]
                          ,(CADDR Item]
         (SETQ SubItemList (CADDDR Item))
         (if (AND SubItemList (EQ (CAR SubItemList)
                                  'SUBITEMS))
             then (SETQ SubItems (CDR SubItemList))
                  [APPEND MenuItem (LIST (ATTACH 'SUBITEMS (for SubItem in SubItems collect (
                                                                    NC.CreateSessionIconNoteFileMenuItem
                                                                        SubItem]
          else MenuItem])
```

(NCP.AddDefaultNoteFileIconMiddleButtonItems

```
[LAMBDA (NewItems)                                          (* pmi%: " 5-Jan-88 15:11")
```

(* * |10/23/87.| dsj. Uniquely add NewItems to NC.DefaultNoteFileIconMiddleButtonItems so that ALL open notefiles will
have the new menu item without needing to be "initialized" first.)

(* * pmi 1/5/88%: Added this function created by DJ (see above comment)%.
Added GLOBALVARS declaration.)

```
    (DECLARE (GLOBALVARS NC.DefaultNoteFileIconMiddleButtonItems))
    (AND NewItems (NCONC NC.DefaultNoteFileIconMiddleButtonItems (for NewItem in NewItems collect NewItem
                                                                    when (NOT (MEMBER NewItem
                                                                                  NC.DefaultNoteFileIconMiddleButtonItems
                                                                                      ])
```

(NCP.NoticedNoteFileNamesMenu

```
[LAMBDA (IncludeNewNoteFileFlg AllowedOperations InterestedWindow Operation)
                                                            (* pmi%: " 9-Dec-87 12:33")
```

```
        (* * AllowedOperations should be one of the atoms%: OPEN, CLOSED, or NIL for both.
        Operation should be a string or atom containing the name of the operation to be performed on the result and the word
        NoteFile; e.g. (QUOTE Open% NoteFile) This is used in the prompt for a new notefile name.)

        (* * dsj 11/6/87%: Give user a menu of noticed notefile names.)

        (* * pmi 12/9/87%: Added this function, which was created by dsj
        (see above comment.))

    (NC.NoticedNoteFileNamesMenu IncludeNewNoteFileFlg AllowedOperations InterestedWindow Operation])
```

(NCP.AddTitleBarMenuItemsToType

```
  [LAMBDA (Type Button NewMenuItems TopOrBottom)                    ; Edited 19-Jan-88 15:51 by Randy.Gobbel

        (* * DSJ%: |10/6/87| Add NewMenuItems to the menu items for the card type if not already a menu item.
        Also use TopOrBottom (one of (QUOTE Top) or (QUOTE Bottom)) to determine where in the menu the new items should be
        placed. If TopOrBottom is NIL put on bottom of menu. Button should be one of
        (QUOTE Left) or (QUOTE Right,) indicating which menu to add to.)

        (* * pmi 12/31/87%: Added DSJ's function (renamed) to the programmer's interface;
        see comment above.)

    (LET (Items Menu)                                               (* If NULL Items then assume we don't want it for this card type)
         (SETQ Menu (SELECTQ (U-CASE Button)
                         (LEFT 'LeftButtonMenuItems)
                         (RIGHT 'RightButtonMenuItems)
                         (NCP.ReportError 'NCP.AddTitleBarMenuItemsToType "Button incorrectly specified.")))
         (SETQ Items (NCP.CardTypeVar Type Menu))
         (AND Items (NCP.ChangeCardTypeFields
                     Type NIL `',(,Menu ,(SELECTQ (OR (U-CASE TopOrBottom)
                                                      'BOTTOM)
                                              (TOP (NCONC (for NewItem in NewMenuItems collect NewItem
                                                             when (NOT (MEMBER NewItem Items)))
                                                          Items))
                                              (BOTTOM [APPEND Items (for NewItem in NewMenuItems collect NewItem
                                                                       when (NOT (MEMBER NewItem Items])
                                              (NCP.ReportError 'NCP.AddTitleBarMenuItemsToType "Location of new
                                                     items in the menu incorrectly specified."])
```

(NCP.AddTitleBarMenuItemsToWindow

```
  [LAMBDA (Window Button NewMenuItems TopOrBottom)                  (* pmi%: "31-Dec-87 13:39")

        (* * pmi 12/31/87%: Add the given menu items to the appropriate button menu of Window, as determined by Button.
        Also use TopOrBottom (one of (QUOTE Top) or (QUOTE Bottom)) to determine where in the menu the new items should be
        placed. If TopOrBottom is NIL put on bottom of menu. Button should be one of
        (QUOTE Left) or (QUOTE Right,) indicating which menu to add to.
        This change was borrowed from dsj's implementation of NCP.AddTitleBarMenuToType.)

    (LET ([Menu (WINDOWPROP Window (SELECTQ (U-CASE Button)
                                    (LEFT 'TitleBarLeftButtonMenu)
                                    (RIGHT 'TitleBarRightButtonMenu)
                                    (NCP.ReportError 'NCP.AddTitleBarMenuItemsToWindow "Button incorrectly
                                           specified."]
          Items)
         (if Menu
             then (SETQ Items (fetch (MENU ITEMS) of Menu))
                  (replace (MENU ITEMS) of Menu with (SELECTQ (OR (U-CASE TopOrBottom)
                                                                  'BOTTOM)
                                                          (TOP (NCONC (for NewItem in NewMenuItems collect NewItem
                                                                         when (NOT (MEMBER NewItem Items)))
                                                                      Items))
                                                          (BOTTOM [APPEND Items
                                                                          (for NewItem in NewMenuItems collect NewItem
                                                                             when (NOT (MEMBER NewItem Items])
                                                          (NCP.ReportError 'NCP.AddTitleBarMenuItemsToWindow
                                                                 "Location of new items in the menu incorrectly
                                                                 specified.")))
                  (replace (MENU IMAGE) of Menu with NIL])
```

(NCP.BringUpSessionIcon

```
  [LAMBDA (IconPosition)                                            (* pmi%: " 9-Dec-87 12:07")

        (* * Start up NoteCards by bringing up the NoteCards icon at IconPosition)

    (NC.BringUpNoteCardsIcon (if (POSITIONP IconPosition)
                                 then IconPosition])
```

(NCP.SessionIconWindow

```
  [LAMBDA NIL                                                       (* pmi%: " 8-Dec-87 15:32")

        (* * pmi 12/8/87%: Created to provide programmer's interface access to NC.NoteCardsIconWindow.)

    (DECLARE (GLOBALVARS NC.NoteCardsIconWindow))
```

```
    NC.NoteCardsIconWindow])

)
```

;;; Miscellaneous

```
(DEFINEQ
```

(NCP.TitleSearch

```
  [LAMBDA (NoteFile Keys CaseSensitiveFlg)                    (* rht%: "11-Nov-86 12:38")
```

(* * Return a list of all cards which contain each string of Args within their titles.)

(* * rht 11/18/85%: Updated to handle new notefile and card object formats.
Note that first arg is now expected to be a notefile.)

(* * kirk 26Jun86 Added BLOCK)

(* * rht 7/12/86%: Cleaned up a bit.)

(* * rht 7/14/86%: Now takes 3 args instead of unlimited number and accepts CaseSensitiveFlg.)

(* * rht 11/11/86%: Took out ENV args to FUNCTION because they seemed to be causing stack overflows.)

```
    (AND (NCP.OpenNoteFileP NoteFile)
         (NCP.MapCards NoteFile [FUNCTION (LAMBDA (Card)
                                            Card]
                (if CaseSensitiveFlg
                    then [FUNCTION (LAMBDA (Card)
                                     (LET ((Title (NCP.CardTitle Card)))
                                          (for Key in (MKLIST Keys) always (STRPOS Key Title]
                  else (FUNCTION (LAMBDA (Card)
                                   (LET ((Title (NCP.CardTitle Card)))
                                        (for Key in (MKLIST Keys) always (STRPOS (U-CASE Key)
                                                                                 (U-CASE Title])
```

(NCP.PropSearch

```
  [LAMBDA Args                                                (* rht%: "12-Jul-86 12:45")
```

(* * Return a list of all IDs which contain each property or property pair appearing in Args.
For each atomic element in Args, there must be a property by that name with non-nil value.
For each pair (list of length 2) in Args, there must be a property and value matching that pair.)

(* * rht 11/18/85%: Updated to handle new notefile and card object formats.
Note that first arg is now expected to be a notefile.)

(* * rht 7/12/86%: Cleaned up a bit.)

```
    (LET ((NoteFile (ARG Args 1)))
         (AND (type? NoteFile NoteFile)
              (NCP.MapCards NoteFile [FUNCTION (LAMBDA (Card)
                                                 Card]
                     (FUNCTION (LAMBDA (Card)
                                 (LET ((PropList (NCP.CardPropList Card))
                                       Prop)
                                      (for i from 2 to Args always (if (ATOM (SETQ Prop (ARG Args i)))
                                                                       then (LISTGET PropList Prop)
                                                                     elseif (AND (LISTP Prop)
                                                                                 (EQ (LENGTH Prop)
                                                                                     2))
                                                                       then (EQ (LISTGET PropList (CAR Prop))
                                                                                (CADR Prop])
```

(NCP.WhichCard

```
  [LAMBDA (WindowOrx y)                                       ; Edited  3-Dec-87 19:01 by rht:
```

(* * Return the ID of the card at a position determined as follows%: If WindowOrx is a position, then use that, if WindowOrx and y are numbers then use (WindowOrx,y) (\, else) use cursor position.)

```
    (NC.CoerceToCard (OR (WINDOWP WindowOrx)
                         (WHICHW WindowOrx y])
```

(NCP.WhichNoteFile

```
  [LAMBDA (WindowOrx y)                                       ; Edited  3-Dec-87 19:01 by rht:
```

(* * Return the notefile corresponding to the window at a position determined as follows%: If WindowOrx is a position, then use that, if WindowOrx and y are numbers then use (WindowOrx,y)
(\, else) use cursor position. Works if the window is either for a card or for a notefile menu.)

```
    (LET [(Window (OR (WINDOWP WindowOrx)
                      (WHICHW WindowOrx y]
         (OR (WINDOWPROP Window 'NoteFile)
             (LET [(Card (WINDOWPROP Window 'NoteCardObject]
```

```
                (AND (NCP.ValidCardP Card)
                     (NCP.NoteFileFromCard Card])
```

(NCP.SelectCards

```
[LAMBDA (InstigatingCardOrWindow SingleCardFlg SelectionPredicate Msg CheckForCancelFlg NewCardFlg)
                                                                    ; Edited 31-Mar-88 15:18 by pmi
```

;; Return a list of cards selected. A menu pops up near the prompt window with 'DONE' and 'CANCEL' buttons. User selects by clicking in card's ;; title bar.

;; rht 11/18/85: Updated to handle new notefile and card object formats. Now takes optional extra args and passes to NC.SelectNoteCards.

;; pmi 12/12/86: Removed obsolete ReturnLinksFlg argument in call to NC.SelectNoteCards.

;; rht 3/2/87: Fix to bug #342: Now makes sure instigating card is displayed, else passes NIL to NC.SelectNoteCards.

;; rg 4/2/87 added NCP.WithLockedCards wrapper

;; pmi 2/26/88: Undid Randy't change of 12/12/86. Now lets NC.CoerceToInterestedWindow (called from NC.SelectNoteCards) check that the ;; instigating card is displayed.

;; pmi 3/31/88: Added NewCardFlg arg. A non-NIL NewCardFlg signals that the menu with the NewCard option should be passed down to ;; NC.SelectCards.

```
(NCP.WithLockedCards (NC.SelectNoteCards SingleCardFlg SelectionPredicate
                           (if SingleCardFlg
                               then (if NewCardFlg
                                        then NC.SelectingSingleCardMenu
                                      else NC.SelectingCardMenu)
                             else NC.SelectingCardsMenu)
                           InstigatingCardOrWindow Msg CheckForCancelFlg])
```

(NCP.DocumentParameters

```
[LAMBDA (NewProps)                                          (* rht%: " 2-Mar-87 22:07")
```

(* * Returns the old value of the MakeDocument default parameters.
If NewProps is non-nil then it should be a prop list which will be used to change some or all of the current MakeDocument parameters. Only those props whose names are valid MakeDocument parameters and whose values are permissible values for that name are used.)

(* * rht 11/6/86%: No longer checks for valid link label since that requires a notefile.
Also returns just a proplist of the params that user can change.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
   (DECLARE (GLOBALVARS NC.MakeDocParameters))
   (LET [(OldParams (for ParamThing in NC.MakeDocParameters unless (EQ (CAR ParamThing)
                                                                       '--DONE--)
                       join (LIST (CAR ParamThing)
                                  (GETPROP 'NC.MakeDocParameters (CAR ParamThing]
        [if NewProps
            then (for Params on NewProps by (CDDR NewProps) bind Param NewValue LegalValues
                    do (SETQ Param (CAR Params))
                       (COND
                          ([NULL (SETQ LegalValues (CDR (FASSOC Param NC.MakeDocParameters]
                           (NCP.ReportError 'NCP.DocumentParameters (CONCAT Param " not a document parameter
                                                                            name.")))
                          ((OR (AND (FMEMB (SETQ NewValue (CADR Params))
                                           LegalValues)
                                    (NEQ NewValue 'Select))
                               (AND (LISTP NewValue)
                                    (FMEMB 'Select LegalValues)))
                           (PUTPROP 'NC.MakeDocParameters Param NewValue))
                          (T (NCP.ReportError 'NCP.DocumentParameters (CONCAT NewValue " is not a permissible
                                                                              value for " Param "."]
        OldParams])
```

(NCP.NoteCardsParameters

```
[LAMBDA (NewParams)                                         (* rht%: " 2-Mar-87 22:07")
```

(* * Returns the old value of the Notecards parameters. If NewParams is non-nil then it should be a prop list which will be used to change some or all of the current Notecards parameters.
Only those props whose names are valid Notecards parameters and whose values are permissible values for that name are used. On NC.NoteCardsParameters's prop list under the parameter name is a list of one or two items.
The first is the name of the global var. The second if present, is a function of no args which returns a list of legal values for that parameter. We only do type checking if that function is present.)

(* * rht 3/20/85%: Changed to use new GLOBALPARAMETER record, especially the CheckFn field.)

(* * rht 6/12/86%: Now checks first whether there's a PARAMCHECKFN before apply*'ing it.)

(* * rht 7/12/86%: Cleaned up a bit.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
   (LET (OldParams PropVal ParamCheckFn)
        [SETQ OldParams (for Param in NC.NoteCardsParameters
                           join (SETQ PropVal (GETPROP 'NC.NoteCardsParameters Param))
```

```
                              (LIST Param (EVAL (if (LISTP PropVal)
                                                    then (fetch (GLOBALPARAMETER PARAMGLOBALVAR) of PropVal)
                                                  else PropVal]
        [if NewParams
            then (for Params on NewParams by (CDDR NewParams) bind Param NewValue GlobalVar PropVal
                    do (if (FMEMB (SETQ Param (CAR Params))
                                  NC.NoteCardsParameters)
                           then (SETQ NewValue (CADR Params))
                                (SETQ PropVal (GETPROP 'NC.NoteCardsParameters Param))
                                (SETQ GlobalVar (if (LISTP PropVal)
                                                    then (fetch (GLOBALPARAMETER PARAMGLOBALVAR) of PropVal)
                                                  else PropVal))
                                (if (OR (ATOM PropVal)
                                        (NULL (SETQ ParamCheckFn (fetch (GLOBALPARAMETER PARAMCHECKFN)
                                                                    of PropVal)))
                                        (APPLY* ParamCheckFn NewValue))
                                    then (SET GlobalVar NewValue)
                                  else (NCP.ReportError 'NCP.NoteCardsParameters (CONCAT NewValue " is not a
                                                                                         permissible value for "
                                                                                         Param "."))))
                         else (NCP.ReportError 'NCP.NoteCardsParameters (CONCAT Param " not a Notecards parameter
                                                                                name."]
        OldParams])
```

(NCP.PrintMsg

```
  [LAMBDA Args                                                    (* rht%: "10-Jul-86 23:43")
```

(* * Expects args of form (<window> <clearFirstFlg> <arg1> <arg2> |...|) and prints the <arg>s to <window>'s prompt window or to the lisp prompt window if <window> is nil. Will clear first if second arg is non-nil.)

```
    (APPLY 'NC.PrintMsg (for i from 1 to Args collect (ARG Args i])
```

(NCP.ClearMsg

```
  [LAMBDA (Window ClosePromptWinFlg WaitMsecs)                    ; Edited 14-Jun-88 14:45 by Randy.Gobbel
```

;;; Clears the prompt window for Window. Will close if ClosePromptWinFlg is non-nil.

```
    ;; rg 6/14/88: Added WaitMsecs arg.
    (NC.ClearMsg Window ClosePromptWinFlg WaitMsecs])
```

(NCP.AskUser

```
  [LAMBDA (Msg Prompt FirstTry ClearFirstFlg MainWindow DontCloseAtEndFlg DontClearAtEndFlg PROMPTFORWORDFlg)
                                                                  (* rht%: "30-May-85 21:49")
```

(* * Asks a question in the prompt window. Just calls the NC.AskUser function.)

```
    (NC.AskUser Msg Prompt FirstTry ClearFirstFlg MainWindow DontCloseAtEndFlg DontClearAtEndFlg
           PROMPTFORWORDFlg])
```

(NCP.AskYesOrNo

```
  [LAMBDA (Msg Prompt FirstTry ClearFirstFlg MainWindow DontCloseAtEndFlg DontClearAtEndFlg)
                                                                  (* rht%: "12-Jul-86 13:15")
```

(* * Asks a yes/no question in the prompt window. Just calls the NC.AskYesOrNo function.)

```
    (NC.AskYesOrNo Msg Prompt FirstTry ClearFirstFlg MainWindow DontCloseAtEndFlg DontClearAtEndFlg])
```

(NCP.RegisterCardByName

```
  [LAMBDA (Name Card RegistryCard)                                (* rht%: "23-May-87 23:21")
```

(* * Hash Card under Name in Card's notefile's system registry card.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

(* * rht 5/23/87%: Added RegistryCard arg which defaults to the notefile registry card.)

```
    [OR RegistryCard (SETQ RegistryCard (fetch (NoteFile RegistryCard) of (NCP.NoteFileFromCard Card]
    (COND
       ((NOT (NC.ValidCardP Card))
        (NCP.ReportError 'NCP.RegisterCardByName (CONCAT Card " is not a valid notecard.")))
       ([NOT (AND (NC.ValidCardP RegistryCard)
                  (EQ (NCP.CardType RegistryCard)
                      'Registry]
        (NCP.ReportError 'NCP.RegisterCardByName (CONCAT RegistryCard " is not a valid registry card.")))
       (T (NC.RegisterCardByName RegistryCard Name Card])
```

(NCP.ListRegisteredCards

```
  [LAMBDA (NoteFileOrRegistryCard IncludeKeysFlg)                 (* pmi%: "10-Dec-87 17:19")
```

(* * Return the list of cards registered in the RegistryCard or notefile.
If IncludeKeysFlg is non-nil, then return dotted pairs of key and card, else just list of cards.)

(* * pmi 12/10/87%: now returns cards instead of UIDs.)

```
(LET (RegistryCard NoteFile Result)
     [COND
        ((NCP.OpenNoteFileP NoteFileOrRegistryCard)
         (SETQ RegistryCard (fetch (NoteFile RegistryCard) of NoteFileOrRegistryCard))
         (SETQ NoteFile NoteFileOrRegistryCard))
        ((AND (NC.ValidCardP NoteFileOrRegistryCard)
              (EQ (NCP.CardType NoteFileOrRegistryCard)
                  'Registry))
         (SETQ RegistryCard NoteFileOrRegistryCard)
         (SETQ NoteFile (NCP.CardNoteFile RegistryCard)))
        (T (NCP.ReportError 'NCP.ListRegisteredCards (CONCAT "Improper arg: " NoteFileOrRegistryCard]
     [if IncludeKeysFlg
         then [MAPHASH (NCP.CardSubstance RegistryCard)
                       (FUNCTION (LAMBDA (Val Key)
                                 (push Result (CONS Key (NC.CardFromUID Val NoteFile]
       else (MAPHASH (NCP.CardSubstance RegistryCard)
                     (FUNCTION (LAMBDA (Val Key)
                                 (push Result (NC.CardFromUID Val NoteFile]
     Result])
```

(NCP.LookupCardByName

```
[LAMBDA (Name NoteFileOrRegistryCard)                    ; Edited 21-Dec-88 15:33 by pmi
```

;; Lookup Name in notefile's system registry card.

;; rht 11/16/86: Changed call to NCP.ReportError

;; rht 5/23/87: Second arg can now be RegistryCard or NoteFile. If the latter, then grab notefile's RegistryCard.

;; pmi 12/21/88: Now returns NIL if the NoteFile does not have a registry card (as when called from NC.OpenNotefile (via NC.CloseNotefile and ;; NC.AbortSession) with partially opened notefile).

```
(LET (RegistryCard)
     [COND
        ((NCP.OpenNoteFileP NoteFileOrRegistryCard)
         (SETQ RegistryCard (fetch (NoteFile RegistryCard) of NoteFileOrRegistryCard)))
        ((AND (NC.ValidCardP NoteFileOrRegistryCard)
              (EQ (NCP.CardType NoteFileOrRegistryCard)
                  'Registry))
         (SETQ RegistryCard NoteFileOrRegistryCard))
        (T (NCP.ReportError 'NCP.LookupCardByName (CONCAT "Improper arg: " NoteFileOrRegistryCard]
```

;; Don't try to call NC.LookupCardByName if RegistryCard is NIL.

```
     (if RegistryCard
         then (NC.LookupCardByName RegistryCard Name])
```

(NCP.UnregisterName

```
[LAMBDA (Name NoteFileOrRegistryCard)                    (* rht%: "23-May-87 23:37")
```

(* * Lookup Name in notefile's system registry card.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

(* * rht 5/23/87%: Second arg can now be RegistryCard or NoteFile.
If the latter, then grab notefile's RegistryCard.)

```
(LET (RegistryCard)
     [COND
        ((NCP.OpenNoteFileP NoteFileOrRegistryCard)
         (SETQ RegistryCard (fetch (NoteFile RegistryCard) of NoteFileOrRegistryCard)))
        ((AND (NC.ValidCardP NoteFileOrRegistryCard)
              (EQ (NCP.CardType NoteFileOrRegistryCard)
                  'Registry))
         (SETQ RegistryCard NoteFileOrRegistryCard))
        (T (NCP.ReportError 'NCP.UnregisterName (CONCAT "Improper arg: " NoteFileOrRegistryCard]
     (NC.UnregisterName RegistryCard Name])
```

(NCP.DisplayedCards

```
[LAMBDA (NoteFiles CardTypes)                            (* pmi%: " 7-Dec-87 16:21")
```

(* * Return a list of all cards that are currently displayed in a window.)

(* * dsj 10/1/87%: Added NoteFiles and CardTypes args. If either are NIL, then all NoteFiles and/or CardTypes are considered.)

(* * pmi 12/7/87%: Integrated above change by dsj into NC sources.)

```
(SETQ NoteFiles (MKLIST NoteFiles))
(SETQ CardTypes (MKLIST CardTypes))
(for Win in (OPENWINDOWS) bind Card when (AND (SETQ Card (NC.CoerceToCard (OR (WINDOWPROP Win 'ICONFOR)
                                                                              Win)))
                                              (if NoteFiles
                                                  then (for NoteFile in NoteFiles
                                                          bind (CardNoteFile _ (NCP.CardNoteFile Card))
```

```
                                                    thereis (EQ NoteFile CardNoteFile))
                                              else T)
                                            (if CardTypes
                                                then (for Type in CardTypes bind (CardType _ (NCP.CardType
                                                                                                 Card))
                                                    thereis (EQ Type CardType))
                                              else T))
        collect Card])
```

(NCP.CardUserDataProp

```
[LAMBDA Args                                                    (* rht%: " 2-Mar-87 22:08")
```

(* * Expects two or three arguments%: Card, Property, and optional new value.
Returns the old value. Assigns the new value if given. Semantics are just like WINDOWPROP.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (LET (Card PropList)
         (COND
            ((AND (NEQ Args 2)
                  (NEQ Args 3))
             (NCP.ReportError 'NCP.CardUserDataProp (CONCAT "Improper number of args: " Args))
             NIL)
            [(NC.ValidCardP (SETQ Card (ARG Args 1)))
             (PROG1 (NC.FetchUserDataProp Card (ARG Args 2))
                 (if (EQ Args 3)
                     then (NC.SetUserDataProp Card (ARG Args 2)
                                  (ARG Args 3))))]
            (T (NCP.ReportError 'NCP.CardUserDataProp (CONCAT Card " not an existing card or box."))
               NIL])
```

(NCP.NoteFileProp

```
[LAMBDA Args                                                    (* rht%: " 2-Mar-87 22:08")
```

(* * Expects two or three arguments%: Notefile, Property, and optional new value.
Returns the old value. Assigns the new value if given. Semantics are just like WINDOWPROP.)

(* * rht 11/16/86%: Changed call to NCP.ReportError)

```
    (LET (NoteFile PropList)
         (COND
            ((AND (NEQ Args 2)
                  (NEQ Args 3))
             (NCP.ReportError 'NCP.NoteFileProp (CONCAT "Improper number of args: " Args))
             NIL)
            [(type? NoteFile (SETQ NoteFile (ARG Args 1)))
             (PROG1 (LISTGET (SETQ PropList (fetch (NoteFile UserProps) of NoteFile))
                            (ARG Args 2))
                 [if (EQ Args 3)
                     then (if PropList
                              then (LISTPUT PropList (ARG Args 2)
                                            (ARG Args 3))
                            else (replace (NoteFile UserProps) of NoteFile with (LIST (ARG Args 2)
                                                                                      (ARG Args 3]))]
            (T (NCP.ReportError 'NCP.NoteFileProp (CONCAT NoteFile " not a valid notefile."))
               NIL])
```

(NCP.SetUpTitleBar

```
[LAMBDA (CardWindow CardType)                                   (* rht%: " 8-Dec-86 18:00")
```

(* * Set up the left and middle button menus for this card. Also handle installation of button event fn.)

```
    (NC.InstallTitleBarLeftMenu CardWindow CardType)
    (NC.InstallTitleBarMiddleMenu CardWindow CardType)
    (NC.InstallTitleBarButtonEventFn CardWindow (FUNCTION NC.TitleBarButtonEventFn])
```

(NCP.AddNoteFileIconMiddleButtonItems

```
[LAMBDA (NoteFile MenuItems)                                    (* pmi%: " 5-Jan-88 15:44")
```

(* * Add the given MenuItems to the given NoteFile.)

(* * pmi 1/5/88%: Now won't add a menu item if it is already on the menu.)

```
    (if NoteFile
        then (LET ((ExistingMenuItems (NCP.NoteFileProp NoteFile 'NoteFileIconMiddleButtonItems))
                   (NoteFileIconWindow (NCP.NoteFileIconWindow NoteFile)))
                  [NCP.NoteFileProp NoteFile 'NoteFileIconMiddleButtonItems
                         (NCONC ExistingMenuItems (for MenuItem in MenuItems collect MenuItem
                                                     when (NOT (MEMBER MenuItem ExistingMenuItems]
```

(* * Smash stashed menu so as to force recompute.)

(* Menu is no longer cached on the window.)
(* (if (OPENWP NoteFileIconWindow) then
(WINDOWPROP NoteFileIconWindow

```
                                                    (QUOTE NoteFileMiddleButtonMenu) NIL)))
                ExistingMenuItems])
```

(NCP.NoteFileIconWindow

```
  [LAMBDA (NoteFile)                                    (* rht%: "23-Nov-86 15:48")

          (* * Return the icon window for given notefile.)

    (WFROMMENU (fetch (NoteFile Menu) of NoteFile])
```

(NCP.CoerceToInterestedWindow

```
  [LAMBDA (WinOrCardOrNoteFile)                         (* pmi%: "21-Oct-87 15:56")

          (* * pmi 10/21/87%: Created to give prog int users the ability to get a reasonable window from various inputs.)

    (NC.CoerceToInterestedWindow WinOrCardOrNoteFile])
```

(NCP.SetGrayShade

```
  [LAMBDA (NewShade)                                    ; Edited 15-Sep-88 15:34 by pmi
    ;; Sets NCP.GrayShade to NewShade.  Also resets the menu of noticed notefiles,  the notefile icon operations menu, and all notefile icon menus.  If
    ;; NewShade is not a texture, then just resets the menus. Returns the new value of NCP.GrayShade.
    (DECLARE (GLOBALVARS NCP.NoticedNoteFileNames NCP.GrayShade))
    (if (TEXTUREP NewShade)
        then (SETQ NCP.GrayShade NewShade))
    (SETQ NC.NoticedNoteFilesMenu NIL)
    (NC.MakeNoteFileIconOperationsMenus)
    (for NoteFileName in NCP.NoticedNoteFileNames bind NoteFile do (REMPROP NoteFileName 'OpenMenuItem)
                                                                   (REMPROP NoteFileName 'OpenMenuItemShaded)
                                                                   (REMPROP NoteFileName 'ClosedMenuItem)
                                                                   (REMPROP NoteFileName 'ClosedMenuItemShaded)
                                                                   (NC.CreateNoteFileMenuItems NoteFileName)
                                                                   (if (SETQ NoteFile (NCP.NoteFileFromFileName
                                                                                        NoteFileName))
                                                                       then (NC.ResetNoteFileInterface NoteFile)))
    NCP.GrayShade])
```

(NCP.MakeTypeIconBitMapSet

```
  [LAMBDA (Bitmap Heights)                              ; Edited 27-Sep-88 11:53 by pmi
    ;; Programmer's interface function for NC.MakeTypeIconBitMapSet.
    ;; Create a prop list of pairs of Height and scaled copies of Bitmap having that height.  If Heights is NIL, NC.DefaultLinkIconAttachedBitMapHeights
    ;; is used.
    (DECLARE (GLOBALVARS NC.DefaultLinkIconAttachedBitMapHeights))
    (NC.MakeTypeIconBitMapSet Bitmap (OR Heights NC.DefaultLinkIconAttachedBitMapHeights])

)

(DECLARE%: DONTEVAL@LOAD

(MOVD 'NCP.WhichCard 'NCP.WC T)

(MOVD 'NCP.WhichNoteFile 'NCP.WNF T)
)


;;; Handy internal functions

(DEFINEQ
```

(NCP.ReportError

```
  [LAMBDA (FunctionName Msg)                            (* rht%: " 2-Mar-87 23:31")

          (* * Print out the various elements of Args to the terminal.)

          (* * rht 11/16/86%: Now takes different args and does optional break.
          Consults NCP.ErrorBrkWhenFlg.)

          (* * rht 3/2/87%: Now coerces FunctionName to atom in case was passed as string.)

    (DECLARE (GLOBALVARS NCP.ErrorBrkWhenFlg))
    (if (EQ NCP.ErrorBrkWhenFlg 'NEVER)
        then (PRIN1 "*** NC-PI error: " T)
             (PRIN1 Msg T)
             (TERPRI T)
      else (APPLY* 'BREAK1 T T (MKATOM FunctionName)
                   NIL NIL (LIST Msg])
```

(NCP.ReportWarning

```
  [LAMBDA (FunctionName Msg)                            (* rht%: "21-Nov-86 00:34")

          (* * Print out the various elements of Args to the terminal.)
```

```
        (* * rht 11/16/86%: Now takes different args and does optional break.
        Consults NCP.ErrorBrkWhenFlg.)

    (DECLARE (GLOBALVARS NCP.ErrorBrkWhenFlg))
    (if (NEQ NCP.ErrorBrkWhenFlg 'ALWAYS)
        then (PRIN1 "*** NC-PI warning: " T)
             (PRIN1 Msg T)
             (TERPRI T)
      else (APPLY* 'BREAK1 T T FunctionName NIL NIL (LIST Msg])
```

(NCP.LinkAnchorDesc

```
  [LAMBDA (Link ToFlg FollowCrossFileLinkFlg)                    (* rht%: "29-Oct-87 18:58")

        (* * Return a description of the anchoring of Link at one of its endpoints.
        The description has the form (<anchormode> <ID> <loc>) If ToFlg is non-nil, then look at the "To" end of the link, otherwise,
        its "From" end.)

        (* * rht 8/25/86%: Now passes non-nil QuietFlg to NCP.UncacheCards.)

        (* * rht 10/29/87%: Now takes FollowCrossFileLinkFlg arg.)

    (LET ((DestCard (fetch (Link DestinationCard) of Link))
          (SrcCard (fetch (Link SourceCard) of Link))
          Card WasActiveP RemoteCard RemoteCrossFileLinkCard)
         (SETQ Card (if ToFlg
                        then (if (AND FollowCrossFileLinkFlg (NC.CrossFileLinkCardP DestCard)
                                      (SETQ RemoteCard (NC.GetCrossFileLinkDestCard DestCard)))
                                 then RemoteCard
                               else DestCard)
                      else (if (AND FollowCrossFileLinkFlg (NC.CrossFileLinkCardP SrcCard)
                                    (SETQ RemoteCard (NC.GetCrossFileLinkDestCard SrcCard))
                                    (SETQ RemoteCrossFileLinkCard (NC.FetchRemoteCrossFileLinkCard SrcCard)))
                               then (PROG1 RemoteCard
                                           (SETQ Link (CAR (NC.GetLinks NIL RemoteCrossFileLinkCard))))
                             else SrcCard)))
         (if (OR (NC.GlobalLinkP Link)
                 ToFlg)
             then (LIST 'GLOBAL Card NIL)
           else (COND
                   ((NOT (SETQ WasActiveP (NC.CardCachedP Card)))
                    (NC.CacheCards Card))
                  (for Obj in (CAR (NC.CollectReferences Card NIL NIL T)) when (NC.SameLinkP Link (CAR Obj))
                     do (COND
                           ((NOT WasActiveP)
                            (NC.UncacheCards Card T)))
                        (RETURN (LIST 'LOCAL Card (CDR Obj])
```

(NCP.GetTypeRecord

```
  [LAMBDA (TypeName)                                             (* rht%: "18-Nov-85 00:40")

        (* * Return the record corresponding to this type name.)

        (* * rht 11/17/85%: Now calls new NC function.)

    (NC.CardTypeRecord TypeName])
```

(NCP.AddLeftButtonTitleBarMenuItems

```
  [LAMBDA (Window NewMenuItems)                                  (* rht%: "12-Jul-86 14:11")

        (* * Add the given menu items to the left button menu of Window.)

    (LET [(Menu (WINDOWPROP Window 'TitleBarLeftButtonMenu]
         (if Menu
             then (PROG1 (replace (MENU ITEMS) of Menu with (APPEND (fetch (MENU ITEMS) of Menu)
                                                                    NewMenuItems))
                         (replace (MENU IMAGE) of Menu with NIL))])
```

(NCP.AddMiddleButtonTitleBarMenuItems

```
  [LAMBDA (Window NewMenuItems)                                  (* rht%: "17-Nov-86 00:15")

        (* * Add the given menu items to the middle button menu of Window.)

    (LET [(Menu (WINDOWPROP Window 'TitleBarMiddleButtonMenu]
         (if Menu
             then (PROG1 (replace (MENU ITEMS) of Menu with (APPEND (fetch (MENU ITEMS) of Menu)
                                                                    NewMenuItems))
                         (replace (MENU IMAGE) of Menu with NIL))])
```

(NCP.CoerceToLinkDisplayMode

```
  [LAMBDA (Thing)                                                (* rht%: "27-May-87 11:28")
```

```
            (* * Thing can be a cardtype, link, atom, list or LINKDISPLAYMODE record.)

            (* * Fixed so that returns non-nil, if Thing is already a linkdisplaymode.)

            (* * rht 5/27/87%: Now checks for case of card type up front.)

    (PROG (DisplayMode)
          (SETQ DisplayMode (COND
                                ((type? LINKDISPLAYMODE Thing)
                                 (RETURN Thing))
                                ((NCP.CardTypeP Thing)
                                 (NCP.CardTypeVar Thing 'LinkDisplayMode))
                                ((NCP.ValidCardP Thing)
                                 (NC.FetchLinkDisplayMode Thing))
                                ((NCP.ValidLinkP Thing)
                                 (NCP.LinkDisplayMode Thing))
                                (T Thing)))
          (SETQ DisplayMode (COND
                                ((type? LINKDISPLAYMODE DisplayMode)
                                 (RETURN DisplayMode))
                                ((FMEMB DisplayMode NCP.LinkDisplayModes)
                                 (NC.MakeNewDisplayMode DisplayMode))
                                ((LISTP DisplayMode)
                                 (create LINKDISPLAYMODE
                                        SHOWTITLEFLG _ (CAR DisplayMode)
                                        SHOWLINKTYPEFLG _ (CADR DisplayMode)
                                        ATTACHBITMAPFLG _ (CADDR DisplayMode)))
                                (T DisplayMode)))
          (RETURN (if (type? LINKDISPLAYMODE DisplayMode)
                      then DisplayMode
                    else NIL])

)


;;; Global variables.

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NCP.ErrorBrkWhenFlg NCP.LinkDisplayModes NCP.TypeFnsAssocLst NCP.NoteCardTypeFns
       NCP.NoteCardTypeVars NC.MakeDocParameters NC.CardTypes NC.SubstanceTypes NC.SystemLinkLabels
       NC.FiledCardLinkLabel NC.SubBoxLinkLabel NC.SelectingCardsMenu NC.SelectingCardMenu
       NC.UCASESystemLinkLabels NC.SourceLinkLabel NC.NoteCardsParameters NCP.NoticedNoteFileNames NCP.GrayShade
       )
)

(RPAQ? NCP.GrayShade GRAYSHADE)

(RPAQ? NCP.NoticedNoteFileNames NIL)

(RPAQ? NCP.DefaultLinkIconAttachedBitMapSize 17)

(RPAQ? NCP.ErrorBrkWhenFlg NIL)

(RPAQ? NCP.LinkDisplayModes '(Icon Title Label Both))

(RPAQ? NCP.NoteCardTypeFns '(MakeFn EditFn QuitFn GetFn PutFn CopyFn MarkDirtyFn DirtyPFn CollectLinksFn
                                DeleteLinksFn UpdateLinkIconsFn InsertLinkFn TranslateWindowPositionFn))

(RPAQ? NCP.NoteCardTypeVars '(SuperType StubFlg FullDefinitionFile LinkDisplayMode DefaultWidth DefaultHeight
                                 LinkAnchorModesSupported DisplayedInMenuFlg LinkIconAttachedBitMap
                                 LeftButtonMenuItems MiddleButtonMenuItems))


;;; Following is for backward compatibility with 1.2

(DECLARE%: DONTEVAL@LOAD

(MOVD 'NCP.OpenCard 'NCP.BringUpCard T)

(MOVD 'NCP.CacheCards 'NCP.ActivateCards T)

(MOVD 'NCP.CardCachedP 'NCP.ActiveCardP T)

(MOVD 'NCP.CardTypeFnP 'NCP.ValidCardTypeFn T)

(MOVD 'NCP.CardTypeP 'NCP.ValidCardType T)

(MOVD 'NCP.CardTypeVarP 'NCP.ValidCardTypeVar T)

(MOVD 'NCP.CloseCards 'NCP.DeactivateCards T)

(MOVD 'NCP.ValidCardP 'NCP.ValidCard T)

(MOVD 'NCP.ContentsFileBox 'NCP.GetContentsFileBox T)

(MOVD 'NCP.OrphansFileBox 'NCP.GetOrphansFileBox T)
```

```
(MOVD 'NCP.ToBeFiledFileBox 'NCP.GetToBeFiledFileBox T)

(MOVD 'NCP.LinkSource 'NCP.GetLinkSource T)

(MOVD 'NCP.LinkDestination 'NCP.GetLinkDestination T)

(MOVD 'NCP.CreateLinkType 'NCP.CreateLinkLabel T)

(MOVD 'NCP.DeleteLinkType 'NCP.DeleteLinkLabel T)

(MOVD 'NCP.RenameLinkType 'NCP.RenameLinkLabel T)

(MOVD 'NCP.LinkTypes 'NCP.GetLinkLabels T)

(MOVD 'NCP.UserLinkTypes 'NCP.GetUserLinkLabels T)

(MOVD 'NCP.ReverseLinkTypes 'NCP.GetReverseLinkLabels T)

(MOVD 'NCP.ValidLinkTypeP 'NCP.ValidLinkLabel T)

(MOVD 'NCP.ValidLinkP 'NCP.ValidLink T)
)

(PUTPROPS NCPROGINT FILETYPE :FAKE-COMPILE-FILE)

(PUTPROPS NCPROGINT MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA NCP.NoteFileProp NCP.CardUserDataProp NCP.PrintMsg NCP.PropSearch NCP.LinkType
                    NCP.LinkDisplayMode NCP.CardRegion NCP.CardSubstance NCP.CardProp NCP.CardTitle
                    NCP.CardTypeDisplayedInMenu)
)
```

FUNCTION INDEX

{MEDLEY}<notecards>system>NCPROGINT.;1

VARIABLE INDEX

PROPERTY INDEX