File created: 16-Jul-2021 12:34:27 **{DSK}<home>frank>il>notecards>system>NCDATABASE.;2**

changes to: (FNS NC.ReadLink)

previous date: 20-May-2021 18:20:36 {DSK}<home>frank>il>notecards>system>NCDATABASE.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ NCDATABASECOMS
       [
```

;;; Internal variables

```
        (GLOBALVARS NC.LockLock NC.SessionProcessInProgress NC.SessionOperationInProgress NC.NoteFileBusyList
               NC.CardBusyList NC.LinksIdentifier NC.ItemIdentifier NC.TitlesIdentifier NC.PropsIdentifier
               NC.IdentifierAtoms NC.DatabaseFileNameSuggestion NC.VersionNumber NC.InitialLinkLabels
               NC.DataFormatVersionNumber NC.DateStringLength NC.ClippedIdentifierAtoms NC.MainMenu
               NC.OrigReadTable NC.TEditPasswordDividedBy100 NC.NoteFileVersionsList NC.NoteFilesHashArray
               NC.NoteFilesHashArraySize NC.UIDCtr NC.LastNoteFileOpened NC.CopyCardsLinksHashArraySize
               NC.CopyBrowserHashArraySize NC.IndexFractionToExpandBy)
        [INITVARS (NC.VersionNumber 3)
               (NC.LinksIdentifier '%###LINKS#)
               (NC.ItemIdentifier '%###ITEM#)
               (NC.TitlesIdentifier '%###TITLES#)
               (NC.PropsIdentifier '%###PROPS#)
               (NC.IdentifierAtoms (LIST NC.LinksIdentifier NC.ItemIdentifier NC.TitlesIdentifier
                                         NC.PropsIdentifier))
               (NC.LastNoteFileOpened NIL)
               (NC.LockLock (CREATE.MONITORLOCK "NoteCards Meta-lock"))
               (NC.SessionProcessInProgress NIL)
               (NC.SessionOperationInProgress NIL)
               (NC.NoteFileBusyList NIL)
               (NC.CardBusyList NIL)
               (NC.OrigReadTable (COPYREADTABLE 'ORIG))
               (NC.NoteFilesHashArraySize 50)
               (NC.UIDCtr 0)
               (NC.CopyCardsLinksHashArraySize 100)
               (NC.CopyBrowserHashArraySize 100)
               (NC.IndexFractionToExpandBy 0.5)
               (NC.DataFormatVersionNumber 1)
               (NC.DateStringLength 18)
               (NC.ClippedIdentifierAtoms (for Atom in NC.IdentifierAtoms collect (SUBATOM Atom 1 -3]
                                                                                  ; Vars for date stuff.
```

;;; The Notecard Database

;;; Implementing the notefile device vector

```
        (FNS NC.CompactNoteFile NC.RemoteHostP NC.DeviceVectorForHost NC.InspectAndRepairNoteFile
             NC.InstallDeviceVectorInNoteFile)
        (GLOBALVARS NC.DeviceVectorsHashArray NC.OpenLocalNoteFilesPublicOrPrivate)
        (INITVARS (NC.DeviceVectorsHashArray (HASHARRAY 10))
               (NC.OpenLocalNoteFilesPublicOrPrivate 'PRIVATE))
```

;;; Creating a NoteFile

```
        (FNS NC.CreateDatabaseFile NC.CreateNoteFile NC.InitializeSpecialCards NC.InitializeSpecialCard)
```

;;; Opening a NoteFile

```
        (FNS NC.NoteFileOpenP NC.OpenDatabaseFile NC.OpenNoteFile NC.CacheTypesAndTitles
             NC.AskUserAboutTruncation NC.InstallCriticalUIDsInNoteFile NC.ProcessInspectAndRepairRequest
             NC.ProcessNoteFileNotFoundError NC.ProcessTruncationRequest NC.ProcessNoteFileNeedsConversionError
             NC.ProcessNoteFileNeedsTruncationError)
        (GLOBALVARS NC.OpenNoteFileFns NC.CloseNoteFileFns)
        (INITVARS NC.OpenNoteFileFns NC.CloseNoteFileFns)
```

;;; Open events card

```
        (FNS NC.RunOpenEvents NC.RunCloseEvents)
        (ADDVARS (NC.OpenNoteFileFns NC.RunOpenEvents)
               (NC.CloseNoteFileFns NC.RunCloseEvents))
```

;;; closing notefiles.

```
        (FNS NC.CloseDatabaseFile NC.CloseNoteFile NC.CloseListOfActiveCards NC.CleanupCardObjects)
        (GLOBALVARS NC.CloseNoteFileFns)
        (INITVARS NC.CloseNoteFileFns)
```

;;; Checkpointing

```
        (FNS NC.CheckpointDatabase NC.CheckpointNoteFile NC.SaveDirtyCards)
```

;;; Aborting an open NoteFile

```
        (FNS NC.AbortSession NC.ForceDatabaseClose)
```

;;; Stuff to handle read-only notefiles.

```
        (FNS NC.ReadOnlyNoteFileP NC.CheckForNotReadOnly)
```

;;; Other database operations.

```
                                                                        ; see also NCCOMPACT
        (FNS NC.DeleteDatabaseFile NC.CopyNoteFile NC.RenameNoteFile NC.RemoveAccessToNoteFile)
```

;;; Functions for getting stuff off the notefile. These manipulate file pointer so run with monitor lock.

```
        (FNS NC.GetNoteCard NC.GetMainCardData NC.GetLinks NC.GetTitle NC.GetPropList NC.GetType
             NC.GetSpecialCards)
        (FNS NC.FetchSpecialCards)
```

;;; Functions for putting stuff on the notefile. These manipulate file pointer so run with monitor lock.

```
        (FNS NC.PutNoteCard NC.PutMainCardData NC.PutLinks NC.PutFromLinks NC.PutRegion NC.PutTitle
             NC.PutPropList NC.PutNoteFileHeader NC.PutCheckptPtr)
```

;;; Functions for reading things off the notefile. Expect file pointer to already be set.

```
        (FNS NC.ReadCardPartHeader NC.ReadIdentifier NC.ReadRegion NC.ReadListOfLinks NC.ReadUID NC.ReadDate
             NC.ReadCardType NC.ReadTitle NC.ReadPropList NC.ReadLink)
```

;;; Functions for writing things on the notefile. Expect file pointer to already be set.

```
        (FNS NC.WriteCardPartHeader NC.WriteIdentifier NC.WriteRegion NC.WriteListOfLinks NC.WriteUID
             NC.WriteDate NC.WriteCardType NC.WriteTitle NC.WritePropList NC.WriteLink)
```

;;; The NoteFile object, Notefiles hash array and accompanying functions.

```
        (FNS NC.NoteFileFromNoteFileUID NC.FetchTopLevelCards NC.StoreNoteFileInHashArray NC.RemoveNoteFile
             NC.TotalIndexSize NC.NoteFileLocFromIndexNum NC.FetchMonitor NC.SetMonitor NC.SameNoteFileP
             NC.ListOfNoteFiles NC.NoteFileFromFileName)
        (FNS NC.RemoveNoteFileFromHashArray NC.RemoveNoteFileName NC.NoticeNoteFile NC.NoticeNoteFileName
             NC.NoticedNoteFileNamesMenu NC.NoteFileNoticedP)
```

;;; Stuff for dealing with the hash array.

```
        (FNS NC.InstallCardInNoteFile NC.CardFromUID)
        (FNS NC.MakeHashKey NC.CreateUIDHashArray)
        (INITVARS (NC.NoteFilesHashArray (NC.CreateUIDHashArray NC.NoteFilesHashArraySize)))
```

;;; Stuff for dealing with CardLocs.

```
        (FNS NC.SetStatus NC.SetMainLoc NC.SetLinksLoc NC.SetTitleLoc NC.SetPropListLoc)
```

;;; The version object.

```
        [INITVARS (NC.NoteFileVersionsList (LIST (create NoteFileVersion Version _ 3 NumberOfReservedCards _ 20
                                                         NoteFileIndexWidth _ 28 NoteFileHeaderSize _ 30]
        (FNS NC.FetchCurrentVersionObject)
```

;;; Stuff for copying cards from one notefile to another, or to the same.

```
        (FNS NC.CopyCards NC.MoveCards NC.PutNoteCardToStream NC.GetNoteCardFromStream NC.MakeHashKeyFromCard)
        (FNS NC.CopyCardPart NC.ExpandIndexInPlace NC.CheckForExpandIndex NC.FindNextCardPart NC.SearchFor###
             NC.RobustReadItemIdentifier NC.RobustReadDate NC.RobustReadUID NC.RobustReadChar NC.RobustReadByte
```

```
        NC.RobustRead NC.CopyCardPartInPlaceToEOF NC.UpdateIndexLocIfNeeded)
    (FNS NC.FixUpLinksInCardCopy NC.FixUpCrossFileLinksInCardCopies NC.FixUpBrowserCardCopy
        NC.BrowserCopyConvertGraphNodeID)
```

;;; Traversal specs, should be in an NCTRAVERSAL module.

```
    (GLOBALVARS NC.TraversalSpecsStylesheet)
    (INITVARS (NC.TraversalSpecsStylesheet (CREATE.STYLE 'ITEMS (LIST (create MENU ITEMS _ T))
                                                         'SELECTIONS
                                                         '(T)
                                                         'ITEM.TITLES
                                                         '(Forward% Links Backward% Links Depth)
                                                         'ITEM.TITLE.FONT
                                                         (FONTCOPY MENUFONT 'WEIGHT 'BOLD)
                                                         'NEED.NOT.FILL.IN
                                                         '(MULTI MULTI NIL)
                                                         'TITLE "Include cards at:")))
    (FNS NC.AskTraversalSpecs)
```

;;; UIDs

```
    (GLOBALVARS NC.UIDBasis NC.GlobalUIDHashArray)
    (INITVARS (NC.UIDBasis NIL))
    (FNS NC.MakeUID NC.LookUpUIDInHashArray NC.InitializeUID NC.SameUIDP)
    (FNS NC.UIDPutProp NC.UIDGetProp NC.UIDSetPropList NC.UIDGetPropList NC.UIDAddProp NC.UIDRemProp)
```

;;; This stuff makes it possible for UIDs encountered by PRINT when writing card's proplists, to be written down in a way that can be read back by READ.

```
    (GLOBALVARS NC.VerticalBarREADTABLE)
    (FNS NC.BuildVerticalBarREADTABLE)
    (INITVARS (NC.VerticalBarREADTABLE (NC.BuildVerticalBarREADTABLE)))
    (FNS NC.ReassembleUID NC.DisassembleUID)
    [DECLARE%: DONTEVAL@LOAD (P (DEFPRINT 'UID (FUNCTION NC.DisassembleUID]
```

;;; contention lock machinery

```
    (FNS NC.NoteFileProp NC.NoteFileAddProp NC.NoteFileDelProp)
    (FNS NC.PrintOperationInProgressMsg NC.CardOperationsInProgress NC.OperationInProgress
        NC.CardCheckOpInProgress NC.NoteFileCheckOpInProgress NC.SessionCheckOpInProgress
        NC.SessionToNoteFileLock NC.LockListOfCards NC.ResetCardProcessInProgress NC.SwitchNoteFileLock)
```

;;; Miscellaneous.

```
    (FNS NC.GetNewCard NC.DatabaseFileName NC.WriteStatus NC.TotalCardsInNoteFile)
    (ADDVARS (HPRINTMACROS (FONTDESCRIPTOR . WRITE.FONTDESCRIPTOR)))
    (FNS WRITE.FONTDESCRIPTOR READ.FONTINTODESCRIPTOR)
    [DECLARE%: DONTEVAL@LOAD (P (NC.StoreAutoloadFnFile (FUNCTION NC.FindNextCardPart)
                                    'NCREPAIR
                                    'NOTECARDSDIRECTORIES]
    (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
          NCDATABASE)
    (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                           (NLAML)
                                                                           (LAMA NC.NoteFileProp])
```

;;; Internal variables

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.LockLock NC.SessionProcessInProgress NC.SessionOperationInProgress NC.NoteFileBusyList
       NC.CardBusyList NC.LinksIdentifier NC.ItemIdentifier NC.TitlesIdentifier NC.PropsIdentifier
       NC.IdentifierAtoms NC.DatabaseFileNameSuggestion NC.VersionNumber NC.InitialLinkLabels
       NC.DataFormatVersionNumber NC.DateStringLength NC.ClippedIdentifierAtoms NC.MainMenu NC.OrigReadTable
       NC.TEditPasswordDividedBy100 NC.NoteFileVersionsList NC.NoteFilesHashArray NC.NoteFilesHashArraySize
       NC.UIDCtr NC.LastNoteFileOpened NC.CopyCardsLinksHashArraySize NC.CopyBrowserHashArraySize
       NC.IndexFractionToExpandBy)
)

(RPAQ? NC.VersionNumber 3)

(RPAQ? NC.LinksIdentifier '%###LINKS#)

(RPAQ? NC.ItemIdentifier '%###ITEM#)

(RPAQ? NC.TitlesIdentifier '%###TITLES#)

(RPAQ? NC.PropsIdentifier '%###PROPS#)

(RPAQ? NC.IdentifierAtoms (LIST NC.LinksIdentifier NC.ItemIdentifier NC.TitlesIdentifier NC.PropsIdentifier))

(RPAQ? NC.LastNoteFileOpened NIL)
```

```
(RPAQ? NC.LockLock (CREATE.MONITORLOCK "NoteCards Meta-lock"))

(RPAQ? NC.SessionProcessInProgress NIL)

(RPAQ? NC.SessionOperationInProgress NIL)

(RPAQ? NC.NoteFileBusyList NIL)

(RPAQ? NC.CardBusyList NIL)

(RPAQ? NC.OrigReadTable (COPYREADTABLE 'ORIG))

(RPAQ? NC.NoteFilesHashArraySize 50)

(RPAQ? NC.UIDCtr 0)

(RPAQ? NC.CopyCardsLinksHashArraySize 100)

(RPAQ? NC.CopyBrowserHashArraySize 100)

(RPAQ? NC.IndexFractionToExpandBy 0.5)

(RPAQ? NC.DataFormatVersionNumber 1)

(RPAQ? NC.DateStringLength 18)

(RPAQ? NC.ClippedIdentifierAtoms (for Atom in NC.IdentifierAtoms collect (SUBATOM Atom 1 -3)))

;; Vars for date stuff.

;;; The Notecard Database

;;; Implementing the notefile device vector

(DEFINEQ

(NC.CompactNoteFile
  [LAMBDA (FromNoteFileOrName ToFileName InPlaceFlg InterestedWindow)
                                                                    ; Edited  8-Dec-88 18:14 by krivacic

;;; Compact a NoteFile.  If InPlaceFlg is T calls NC.CompactNoteFileInPlace.  Otherwise if ToFileName is NIL, asks for a new file name.

;;; fkr 11/8/85 Updated to handle new CardID scheme and NoteFile object.

;;; kirk 19Nov85: Created from NC.CompactDatabaseInPlace to handle new NoteFile format

;;; fgh 5/186 Totally rewritten to get rid of numerous bugs.  Added new InterestedWindow parameter.

;;; rht 7/2/86: Fixed bug in call to NC.CompactToTarget and NC.CompactInPlace.  They were being called with FromNoteFile instead of (OR
;;; FromNoteFile FromFileName).

;;; kirk 3Jul86 Added SETQ NC.DatabaseFileNameSuggestion

;;; fgh 9/1/86 Now just a wrapper that calls the device specific compact fn.  Old CompactNoteFile is now NCLocalDevice.CompactNoteFile.

;;; pmi 12/19/86 Added test for open notefile so we can abort if it is open.  Made consistent with other NoteFile operations in the way it checks for valid
;;; NoteFile, gets msg window, etc.

;;; pmi 5/29/87: Added call to NC.NoticeNoteFile to make sure this NoteFile is noticed.  Cleaned up case where notefilename is valid, but a notefile
;;; object does not exist.  Now creates an interface menu for the target of compaction if NC.NoteFileMenuLingerFlg is T.  This menu for the new notefile
;;; is positioned slightly offset from the original notefile's menu.

;;; pmi 8/13/87: Moved call to NC.NoticeNoteFile for the new notefile (for compact to target) to this function from NC.CompactNoteFileToTarget.

    (DECLARE (GLOBALVARS NC.MsgDelay NC.NoteFileMenuLingerFlg NC.NoteCardsIconWindow))
    (PROG ((MsgWindow InterestedWindow)
           FromNoteFile FromFileName FullFromFileName ReturnValue Menu MenuWindowRegion NewMenu)

;;; Get the name of the file to be compacted

          (if (type? NoteFile FromNoteFileOrName)
              then (SETQ FromNoteFile FromNoteFileOrName)
                   (SETQ FromFileName (fetch (NoteFile FullFileName) of FromNoteFile))
            elseif (SETQ FromFileName (OR FromNoteFileOrName (NC.DatabaseFileName "Name of NoteFile to be
                                                                  compacted:" " -- " T NIL NIL MsgWindow)))
            else (RETURN NIL))

;;; Check for existence of file to be compacted.

          (if (SETQ FullFromFileName (FULLNAME FromFileName))
              then [if (OR FromNoteFile (SETQ FromNoteFile (NC.NoteFileFromFileName FullFromFileName)))
                       then (SETQ MsgWindow (OR MsgWindow (NC.CoerceToInterestedWindow FromNoteFile]
            else (NC.RemoveAccessToNoteFile FromFileName)
                 (SETQ MsgWindow (NC.CoerceToInterestedWindow MsgWindow))
                 (NC.PrintMsg MsgWindow T FromFileName " does not exist." (CHARACTER 13)
                        "Compact cancelled."
```

```
                    (CHARACTER 13))
                (DISMISS NC.MsgDelay)
                (NC.ClearMsg MsgWindow T)
                (RETURN NIL))
```

;;; Check to see if the notefile is open, abort if it is.

```
        (if (NC.NoteFileOpenP FromNoteFile)
            then (NC.PrintMsg MsgWindow T "Can't compact an open notefile." (CHARACTER 13)
                         "Compact cancelled."
                         (CHARACTER 13))
                 (DISMISS NC.MsgDelay)
                 (NC.ClearMsg MsgWindow T)
                 (RETURN NIL))
```

;;; Apply the device specific comnpact notefile fn for the file's host.

```
        (if (SETQ ReturnValue (APPLY* (fetch (NoteFileDevice CompactNoteFileFn)
                                             of (NC.DeviceVectorForHost (FILENAMEFIELD FullFromFileName 'HOST)
                                                                        'PRIVATE))
                                      FullFromFileName ToFileName InPlaceFlg MsgWindow))
            then
```

;;; Add the From notefile to the Hash Array and the list of noticed notefiles, in case it isn't already there.

```
                 (NC.NoticeNoteFile FullFromFileName)
                 [if (NULL InPlaceFlg)
                     then
```

;;; Add the To notefile to the Hash Array and the list of noticed notefiles.

```
                          (NC.NoticeNoteFile ReturnValue)
                          (if NC.NoteFileMenuLingerFlg
                              then (if [SETQ Menu (OR (AND (type? NoteFile FromNoteFile)
                                                           (fetch (NoteFile Menu) of FromNoteFile))
                                                      (GETPROP FullFromFileName 'Menu]
                                       then (SETQ MenuWindowRegion (WINDOWPROP (WFROMMENU Menu)
                                                                               'REGION))
                                            (SETQ NewMenu (NC.SetUpNoteFileInterface ReturnValue
                                                                                     (create POSITION
                                                                                             XCOORD _ (fetch (REGION LEFT)
                                                                                                        of MenuWindowRegion)
                                                                                             YCOORD _ (fetch (REGION BOTTOM)
                                                                                                        of MenuWindowRegion))
                                                                                     MsgWindow))
                                            (RELMOVEW NewMenu (create POSITION
                                                                      XCOORD _ 6
                                                                      YCOORD _
                                                                      (MINUS (PLUS (FONTHEIGHT
                                                                                     (fetch (MENU MENUTITLEFONT)
                                                                                       of Menu))
                                                                                   6]
                            else (NC.PrintMsg MsgWindow T "Compact cancelled." (CHARACTER 13))
                                 (DISMISS NC.MsgDelay)
                                 (NC.ClearMsg MsgWindow T))
                          (RETURN ReturnValue])
```

(NC.RemoteHostP

```
  [LAMBDA (PATTERN HOSTNAME)                                    (* fgh%: " 1-Sep-86 19:23")
```

(* * Returns T if the file pattern is to be located on a NoteCardsServer.)

(* * fgh |9/1/86| Put in checks for DSKNN for Dorado's and for CORE files.)

```
    (PROG ([HOST (OR HOSTNAME (FILENAMEFIELD PATTERN 'HOST]
           ADDRESS)
           (RETURN (AND (NEQ HOST 'DSK)
                        [NOT (AND (EQ (SUBATOM HOST 1 3)
                                      'DSK)
                                  (FIXP (SUBATOM HOST 4]
                        (NEQ HOST 'CORE)
                        (GETD 'NCCLIENT.AREYOUTHERE)
                        (SETQ ADDRESS (LOOKUP.NS.SERVER HOST))
                        (NCCLIENT.AREYOUTHERE ADDRESS])
```

(NC.DeviceVectorForHost

```
  [LAMBDA (Host PublicOrPrivate)                                ; Edited 31-Dec-87 11:49 by Trigg
```

;; Find the appropriate device vector for Host.

;; fgh 9/1/86 First created.

;; rht 12/31/87: Ripped out check for ns host in PRIVATE case. Opening notefiles on ns hosts is okay these days.

```
    (DECLARE (GLOBALVARS NC.DeviceVectorsHashArray NC.OpenLocalNoteFilesPublicOrPrivate))
    (if (NC.RemoteHostP NIL Host)
        then (GETHASH 'REMOTEMULTIUSER NC.DeviceVectorsHashArray)
```

```
      elseif [OR (AND PublicOrPrivate (EQ PublicOrPrivate 'PRIVATE))
                 (AND (NULL PublicOrPrivate)
                      (EQ NC.OpenLocalNoteFilesPublicOrPrivate 'PRIVATE]
        then (GETHASH 'LOCALSINGLEUSER NC.DeviceVectorsHashArray)
      else (GETHASH 'LOCALMULTIUSER NC.DeviceVectorsHashArray])
```

(NC.InspectAndRepairNoteFile

```
  [LAMBDA (NoteFileOrFileName ReadSubstancesFlg InterestedWindow)
                                                                    ; Edited 21-Dec-88 16:25 by pmi
```

;; Check to be sure file is closed before calling real inspect and repair.

;; rht 7/16/86: Added InterestedWindow arg. Removed call to NC.OpenDatabaseFile.

;; rht 7/17/86: Now works with file name args as well as notefile args. Took out reopen'ing of notefile, because you don't know how it was originally ;; opened.

;; fgh 9/1/86 Now just a wrapper for the device specific inspect & repair fn. Old functionality is in NCLocalDevice.InspectAndRepairNoteFile.

;; pmi 12/19/86 Added test for open notefile so we can abort if it is open. Made consistent with other NoteFile operations in the way it checks for ;; valid NoteFile, gets msg window, etc.

;; pmi 5/29/87: Added call to NC.NoticeNoteFile to make sure this NoteFile is noticed. Cleaned up case where notefilename is valid, but a notefile ;; object does not exist.

;; pmi 12/21/88: Now passes either the notefile or its filename to the device-specific repair notefile fn.

```
  (DECLARE (GLOBALVARS NC.MsgDelay))
  (PROG ((MsgWindow InterestedWindow)
         NoteFile FileName FullFileName)
```

;; Get the name of the file to be inspected.

```
        (if (type? NoteFile NoteFileOrFileName)
            then (SETQ NoteFile NoteFileOrFileName)
                 (SETQ FileName (fetch (NoteFile FullFileName) of NoteFile))
          elseif (SETQ FileName (OR NoteFileOrFileName (NC.DatabaseFileName "Name of NoteFile to be inspected and
                                                                            repaired:" " -- " T NIL NIL MsgWindow)))
          else (RETURN NIL))
```

;; Check for existence of file to be inspect and repaired.

```
        (if (SETQ FullFileName (FULLNAME FileName))
            then [if (OR NoteFile (SETQ NoteFile (NC.NoteFileFromFileName FullFileName)))
                     then (SETQ MsgWindow (OR MsgWindow (NC.CoerceToInterestedWindow NoteFile]
          else (NC.RemoveAccessToNoteFile FileName)
               (SETQ MsgWindow (NC.CoerceToInterestedWindow MsgWindow))
               (NC.PrintMsg MsgWindow T FileName " does not exist." (CHARACTER 13)
                      "Inspect & Repair cancelled."
                      (CHARACTER 13))
               (DISMISS NC.MsgDelay)
               (NC.ClearMsg MsgWindow T)
               (RETURN NIL))
```

;; Check to see if the notefile is open, abort if it is.

```
        (if (NC.NoteFileOpenP FullFileName)
            then (NC.PrintMsg MsgWindow T "Can't inspect and repair an open notefile." (CHARACTER 13)
                      "Inspect & Repair cancelled."
                      (CHARACTER 13))
                 (DISMISS NC.MsgDelay)
                 (NC.ClearMsg MsgWindow T)
                 (RETURN NIL))
```

;; Add the From notefile to the Hash Array and the list of noticed notefiles, in case it isn't already there.

```
        (NC.NoticeNoteFile FullFileName)
```

;; Apply the device specific repair notefile fn for the file's host.

```
        (RETURN (APPLY* (fetch (NoteFileDevice RepairNoteFileFn) of (NC.DeviceVectorForHost (FILENAMEFIELD
                                                                                             FullFileName
                                                                                             'HOST)
                                                                                  'PRIVATE))
                        (OR NoteFile FullFileName)
                        ReadSubstancesFlg MsgWindow]))
```

(NC.InstallDeviceVectorInNoteFile

```
  [LAMBDA (NoteFile PublicOrPrivate)                    (* Randy.Gobbel " 2-Feb-87 17:07")
```

(* * Figure out the appropriate device vector for NoteFile from its FullName and install in the device vector slot.)

(* * fgh |5/23/86| First created.)

(* * kef 7/23/86%: Added the GETD NCCLIENT.AREYOUTHERE expression so this function can run without the aid of the NCCLIENT software.)

(* * kef 7/23/86%: Checks at the end to ensure that some sort of NoteFileDevice was installed.)

(* * fgh |9/1/86| Adapted to use NC.DeviceVectorForHost.)

(* * rg |2/2/87| Removed unused NSAddress var)

```
    (LET (Host)
```

```
        (SETQ Host (FILENAMEFIELD (fetch (NoteFile FullFileName) of NoteFile)
                                  'HOST))
        (replace (NoteFile Device) of NoteFile with (NC.DeviceVectorForHost Host PublicOrPrivate))
        (OR (type? NoteFileDevice (fetch (NoteFile Device) of NoteFile))
            (NCP.ReportError "Error in determining device vector for NoteFile"])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.DeviceVectorsHashArray NC.OpenLocalNoteFilesPublicOrPrivate)
)

(RPAQ? NC.DeviceVectorsHashArray (HASHARRAY 10))

(RPAQ? NC.OpenLocalNoteFilesPublicOrPrivate 'PRIVATE)
```

;;; Creating a NoteFile

```
(DEFINEQ
```

(NC.CreateDatabaseFile

```
  [LAMBDA (FileName HashArraySize CallingOperationMsg OmitFinalNoteFlg StartingNextFreeIndex NoSpecialCardsFlg
                 InterestedWindow QuietFlg PublicOrPrivate OpenFlg ReadOnlyFlg Don'tCreateInterfaceFlg
                 MenuPosition)                                          (* pmi%: "15-May-87 19:03")
```

(* * Create a NoteCards database on file FileName. Just creates an index HashArraySize entries long, then writes out the Root and Orphan cards)

(* * rht 8/7/84%: Added OmitFinalNoteFlg flag parameter to prevent the final message.
Changed parameter name from NC.IndexSizeInEntries to HashArraySize since the fomer is a global.)

(* * rht 1/30/85%: Reserved 3 bytes of the remaining 8 to hold pointer to last checkpointed EOFPTR.)

(* * rht 3/21/85%: Added the StartingNextFreeIndex argument which if non-nil, gives a NextID Num to be filled in to the file before returning. This is useful when compacting.)

(* * fkr 11/8/85%: Added check that OPENSTREAM succeeded.
Added call to NC.CreateNoteFileObject in which lots of work is now being done.)

(* * fgh |11/17/85| Wrapped whole thing in ERSETQ to close file if somethinghappens during the create.)

(* * kirk 26Dec85 Added NoSpecialCardsFlg flag for use by Compact to target file.)

(* * rht 1/8/86%: Now reuses old notefile object if there is one for this file name.)

(* * rht 7/13/86%: Added QuietFlg arg.)

(* * fgh |9/1/86| Now just a wrapper for NC.CreateNoteFile to maintain compatibility with callers.
Result of device vector implementation.)

(* * pmi 5/15/87%: Added PublicOrPrivate, OpenFlg, ReadOnlyFlg, Don'tCreateInterfaceFlg, and MenuPosition arguments to correspond to changes in NC.CreateNoteFile.)

```
    (NC.CreateNoteFile FileName HashArraySize NoSpecialCardsFlg InterestedWindow CallingOperationMsg QuietFlg
          PublicOrPrivate OpenFlg ReadOnlyFlg Don'tCreateInterfaceFlg MenuPosition])
```

(NC.CreateNoteFile

```
  [LAMBDA (NoteFileOrFileName SizeInCards Don'tCreateSpecialCards InterestedWindow OperationMsg QuietFlg
                 PublicOrPrivate OpenFlg ReadOnlyFlg Don'tCreateInterfaceFlg MenuPosition)
                                                                        (* pmi%: "24-Jun-87 17:24")
```

(* * Create a NoteFile. Most of the work should be done by the device specific create notefile fn.)

(* * fgh |9/1/86| First created.)

(* * fgh&rht 9/5/86%: Now creates small temporary hash array.)

(* * pmi 5/20/87%: Added call to NC.SetUpNoteFileInterface to create a closed NoteFile Icon after creating the NoteFile.
Now asks if next version should be created if file already exists.)

(* * pmi 6/24/87%: Had to strip version number from filename even when it doesn't exist, in case it came in with the version number of a non-existent file.)

```
    (DECLARE (GLOBALVARS NC.MsgDelay))
    (PROG (NoteFile NoteFileName NoteFileFullName ReturnValue)
```

(* * Get the name from the user if necessary.)

```
          [SETQ NoteFileName (if (type? NoteFile NoteFileOrFileName)
                                 then (fetch (NoteFile FullFileName) of NoteFileOrFileName)
                               else (OR NoteFileOrFileName (NC.DatabaseFileName "What is the name of the file to
                                                                    be created?" " -- " T T NIL InterestedWindow]
          (if (NULL NoteFileName)
              then (RETURN 'CreateCancelled))
```

```
(* * Check to see if a file by this name already exists.)

[if (FILENAMEFIELD NoteFileName 'VERSION)
    then

(* * A version has been specified -
make sure it does not already exist.)

        (if (SETQ NoteFileFullName (FULLNAME NoteFileName 'OLD))
            then

(* * This file already exists as this version.)

(* * Notify user)

                (NC.PrintMsg InterestedWindow T "NoteFile " NoteFileFullName " already exists."
                       (CHARACTER 13))

(* * If the user wants to create the file, then create the next version of it.)

                (SETQ NoteFileFullName (FULLNAME (PACKFILENAME 'VERSION NIL 'BODY NoteFileFullName)
                                                 'NEW))
                (if (NC.AskYesOrNo (CONCAT "Do you want to create " NoteFileFullName " (next available
                                                version)?")
                          " -- " "N" NIL InterestedWindow T NIL)
                  else (NC.PrintMsg InterestedWindow NIL "Create cancelled." (CHARACTER 13))
                       (DISMISS NC.MsgDelay)
                       (NC.ClearMsg InterestedWindow T)
                       (RETURN 'CreateCancelled))
          else

(* * This is already a valid new full file name; use it.)

              (SETQ NoteFileFullName NoteFileName))
  else

(* * No version specified, use the next available one.)

      (SETQ NoteFileFullName (FULLNAME NoteFileName 'NEW]

(* * Create a NoteFile object with a UID, etc.)

[SETQ NoteFile (if (type? NoteFile NoteFileOrFileName)
                   then NoteFileOrFileName
                 else (OR (NC.NoteFileFromFileName NoteFileFullName)
                          (create NoteFile]
(replace (NoteFile UID) of NoteFile with (NC.MakeUID))
(replace (NoteFile MonitorLock) of NoteFile with (CREATE.MONITORLOCK 'Creating% NoteFile))
(replace (NoteFile FullFileName) of NoteFile with NoteFileFullName)
(replace (NoteFile ReadOnlyFlg) of NoteFile with NIL)

(* * only a small hash array for creating a file.)

[replace (NoteFile HashArray) of NoteFile with (NC.CreateUIDHashArray (CONSTANT (LENGTH
                                                                                  (RECORDFIELDNAMES
                                                                                   'NoteFileCriticalUIDs]

(* * Install the appropriate device vector)

(NC.InstallDeviceVectorInNoteFile NoteFile PublicOrPrivate)

(* * Say something to the user.)

(OR QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                  "Creating NoteFile " NoteFileFullName ".  Please wait...  "))

(* * Call the device specific create notefile fn.)

(if [type? NoteFile (SETQ ReturnValue (CAR (ERSETQ (APPLY* (fetch (NoteFile CreateNoteFileFn)
                                                                  of NoteFile)
                                                           NoteFile SizeInCards InterestedWindow
                                                           OperationMsg QuietFlg]
    then

(* * Device specific Create NoteFile fn returned okay. Go an an create the special cards.)

        (if Don'tCreateSpecialCards
          else (replace (NoteFile NextIndexNum) of NoteFile with 1)
               (NC.InitializeSpecialCards NoteFile))

(* * Checkpoint the NF, then close it and return the NF objet.)

        (NC.CheckpointNoteFile NoteFile QuietFlg T InterestedWindow OperationMsg)
        (if [type? NoteFile (SETQ ReturnValue (CAR (ERSETQ (APPLY* (fetch (NoteFile CloseNoteFileFn)
                                                                      of NoteFile)
                                                                NoteFile SizeInCards InterestedWindow
```

```
                                                                    OperationMsg QuietFlg]
                    then                                            (* Close went okay.)
                         (if (NULL QuietFlg)
                             then (NC.PrintMsg InterestedWindow NIL "Done!" (CHARACTER 13))
                                  (NC.ClearMsg InterestedWindow T))
                                                                    (* Clean out the NF object and "notice it" %.)
                         (create NoteFile smashing NoteFile UID _ (fetch (NoteFile UID) of NoteFile)
                                                   FullFileName _ (fetch (NoteFile FullFileName) of NoteFile)
                                                   Menu _ (fetch (NoteFile Menu) of NoteFile)
                                                   NoteFileDevice _ (fetch (NoteFile NoteFileDevice) of NoteFile))
```

(* * If needed, create a closed NoteFile interface.)

```
                         (if (AND (NULL Don'tCreateInterfaceFlg)
                                  (NULL OpenFlg))
                             then (NC.SetUpNoteFileInterface NoteFile MenuPosition InterestedWindow))
```

(* * Notice the notefile)

```
                         (NC.NoticeNoteFile NoteFile)
```

(* * Open the notefile, if requested)

```
                         (if OpenFlg
                             then (NC.OpenNoteFile NoteFile NIL T NIL NIL NIL Don'tCreateInterfaceFlg NIL
                                                   InterestedWindow PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg
                                                   NIL T))
                         (SETQ NC.DatabaseFileNameSuggestion (PACKFILENAME 'VERSION NIL 'BODY
                                                                    (fetch (NoteFile FullFileName) of NoteFile)
                                                                    ))
                                                                    (* Set the DatabaseFileNameSuggestion)
                         (RETURN NoteFile)                          (* Return the NoteFile.)
                    else                                            (* Problems with closing the NoteFile --
                                                                    error)
                        (NC.ReportError "NC.CreateNoteFile" (CONCAT "Could not close notefile after it was
                                                                    created because: " ReturnValue))
                        (RETURN ReturnValue))
               else
```

(* * There was an error in the device specific create notefile fn.)

```
                (NC.ReportError "NC.CreateNoteFile" (CONCAT "Could not create NoteFile due to " ReturnValue "
                                                            error."))
                (RETURN ReturnValue])
```

(NC.InitializeSpecialCards

```
  [LAMBDA (NoteFile)                                                (* rht%: "26-Feb-86 11:45")
```

(* Create and put the initial versions of Root, Orphan, and Unclassified cards onto database specified by DatabaseStream. Also initialize the List of link labels)

(* * rht 11/10/85%: Updated to handle new NoteFile and Card scheme.)

(* * rht 11/14/85%: Now sticks top level cards into NoteFile object.)

(* * rht 2/26/86%: Added new special card, the RegistryCard.)

```
                                                                    (* Contents card)
    (replace (NoteFile TableOfContentsCard) of NoteFile with (NC.InitializeSpecialCard 'FileBox NoteFile "Table of
                                                                      Contents"))
                                                                    (* Orphans card)
    (replace (NoteFile OrphansCard) of NoteFile with (NC.InitializeSpecialCard 'FileBox NoteFile "Orphans"))
                                                                    (* ToBeFiled Card)
    (replace (NoteFile ToBeFiledCard) of NoteFile with (NC.InitializeSpecialCard 'FileBox NoteFile "To Be Filed"))
                                                                    (* Link Labels)
    (replace (NoteFile LinkLabelsCard) of NoteFile with (NC.InitializeSpecialCard 'List NoteFile "Link Labels"
                                                                    NC.InitialLinkLabels))
    (replace (NoteFile RegistryCard) of NoteFile with (NC.InitializeSpecialCard 'Registry NoteFile "Registry"))
   NoteFile])
```

(NC.InitializeSpecialCard

```
  [LAMBDA (CardType NoteFile Title TypeSpecificArgs)                (* kirk%: "27-Nov-85 13:04")
```

(* * Make a new card and write down its card parts.)

```
    (LET ((Card (NC.MakeNoteCard CardType NoteFile Title T TypeSpecificArgs)))
         (NC.PutMainCardData Card)
         (NC.PutTitle Card)
         (NC.PutPropList Card)
         (NC.PutLinks Card)
         (NC.DeactivateCard Card T)
         Card])

)
```

;;; Opening a NoteFile

```
(DEFINEQ
```

(NC.NoteFileOpenP

```
  [LAMBDA (NoteFileOrFileName)                                    (* rht%: "14-Nov-86 21:49")

          (* * Is NoteFile an open NoteFile)

          (* * kef 7/21/86%: Now just calls the device vector function.)

          (* * DVN |11/13/86| now checks to see that NoteFile is a true notefile rather than just a filename.)

          (* * rht 11/14/86%: Changed arg name and syntax. Functionality should be the same.)

    (LET [(NoteFile (COND
                        ((type? NoteFile NoteFileOrFileName)
                         NoteFileOrFileName)
                        ((INFILEP NoteFileOrFileName)
                         (NC.NoteFileFromFileName NoteFileOrFileName]
         (AND (type? NoteFile NoteFile)
              (APPLY* (fetch (NoteFile NoteFileOpenPFn) of NoteFile)
                     NoteFile])
```

(NC.OpenDatabaseFile

```
  [LAMBDA (NoteFileOrFileName Access Don'tCacheTypesAndTitlesFlg QuietFlg Don'tCreateFlg Convertw/oConfirmFlg
                Don'tCreateArrayFlg Don'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg
                InterestedWindow BringUpTOC)                      ; Edited 23-Feb-89 11:03 by krivacic
```

;;; bk 2/23/89: Add NC.BringUpTOC parm to NC.OpenNoteFile

;;; Removed Hash Array argument to NC.OpenNoteFile .

```
    (NC.OpenNoteFile NoteFileOrFileName Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg
           Don'tCreateArrayFlg Don'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg InterestedWindow
           NIL NIL QuietFlg (if (EQ Access 'INPUT)
                                then T
                              else NIL)
           NIL NIL BringUpTOC])
```

(NC.OpenNoteFile

```
  [LAMBDA (NoteFileOrFileName Don'tCacheTypesAndTitlesFlg Don'tCreateFlg ConvertNoConfirmFlg Don'tCreateArrayFlg
                Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg InterestedWindow
                PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg
                Don'tCheckVersionFlg BringUpTOC)                  ; Edited 23-Feb-89 10:59 by krivacic
```

;;; fgh 5/23/86 Renamed to NC.OpenNoteFile from NC.OpenDatabaseFile. Total revamp to implement device vector.

;;; kef 7/18/86: Inserted a call to stuff the UID into the NoteFile because BuildHashArray needed it.

;;; kef 7/21/86: Moved up the install of the NoteFile into the NoteFileHashArray to before the building of the NoteFile's hash array. The reason is that the
;;; remote multi client build hash array function needs to get a list of UIDs, and in order to do this, it needs to grab a Courier stream for the NoteFile given
;;; only the UID. It can only do this if the UID is registered in the NoteFilesHashArray.

;;; fgh 8/31/86 Updated to account for changes made to system since 5/23/86 revamp. Changes reimplemented include: (fgh 6/8/86 Added code to
;;; insure that two files with SameUIDP would never be open at once.) (fgh 6/25/86 Added contention locks -- NC.ProtectedNoteFileOperation,
;;; Don'tCheckOperationInProgressFlg etc.) (fgh 6/27/86 Added MenuPsotion arg to pass to SetUpNoteFileInterface) (kirk 15Jul86 Added call to
;;; NC.SetUpNoteFileInterface if already open)

;;; fgh 9/1/86 Reimplemented ReadOnly NoteFile open.

;;; fgh 9/4/86 Put in default for NoteFilesHashArray which is NC.NoteFilesHashArray

;;; kirk/rht 8/29/86: Now resets Name after conversion from version 2 to version3.

;;; rht 10/29/86: Changed 'aborted' to 'canceled' in message.

;;; rht 10/31/86: Added Don'tCheckForTruncationFlg arg.

;;; rht&pmi 11/21/86: Took away the protection from around the AFTER call to open events fns.

;;; pmi 12/12/86: Added InterestedWindow argument to NC.SetUpNoteFileInterface so that it can print a prompt to the user about placing the NoteFile
;;; menu.

;;; rg 3/4/87 Added NC.ProtectedSessionOperation wrapper, removed Don'tCheckOperationsInProgressFlg

;;; rht 3/25/87: Now calls NC.CoerceToInterestedWindow.

;;; pmi 3/31/87: Moved line of code which sets the ReadOnlyFlg to just after the test for an open notefile. Otherwise, a notefile opened read-only could
;;; be changed to one opened normally.

;;; rht 4/2/87: Now passes InterestedWindow to opennotefilefns.

;;; rg 4/2/87 enlarged scope of NC.ProtectedNoteFileOperation

```
;;; RG 4/3/87 replaced missing InterestedWindow arg to OpenNoteFileFn

;;; pmi 5/19/87: Removed NoteFilesHashArray argument.  Replaced call to NC.StoreNoteFileInHashArray with NC.NoticeNoteFile in general cleanup.

;;; pmi 5/20/87: Moved the open test up to almost the beginning of the function.

;;; pmi 5/29/87: Deleted extra InterestedWindow argument to NC.ProtectedNoteFileOperation.  Added call to NC.RemoveAccessToNoteFile to 'unnotice'
;;; this notefile if the file does not exist, and remove its icon, if it has one.  If InterestedWindow is the window for this NoteFile's interface, then change it
;;; to NC.NoteCardsIconWindow.

;;; pmi 6/3/87: Added check and warning for filenames which do not have the .notefile extension and which have not yet been noticed (operated on) by
;;; NoteCards.

;;; rht&pmi 6/4/87: Added TempInterestedWindow var to use until real InterestedWindow can be computed.

;;; rht 6/8/87: Fixed what happens for notefiles with bad headers.

;;; pmi 6/24/87: Added '(Highest version)' to question about opening highest version of a file.  Added Don'tCheckVersionFlg for Create and Compact,
;;; which have already figured out the correct version.

;;; pmi 12/17/87: Added Don'tCreateInterfaceFlg argument to NC.SetUpNoteFileInterface so that esisting notefile icons will be updated properly when the
;;; Don'tCreateInterfaceFlg is T, as suggested by dsj.

;;; bk 2/23/89: Added the BringUpTOC parameter to bring up the TOC on a successfull open.

    (DECLARE (GLOBALVARS NC.OpenNoteFileFns NC.LastNoteFileOpened))

;;; NOTE: Session lock turns into NoteFile lock after NoteFile is created

    (PROG ((TempInterestedWindow (OR InterestedWindow (NC.CoerceToInterestedWindow NoteFileOrFileName)))
           NoteFile FileName NewerFileName OldVersion NewVersion NoteFileMenu ReturnValue CriticalUIDs)

;;; Figure out the name of the file containing the NoteFile

        (if [NULL (SETQ FileName (if (type? NoteFile NoteFileOrFileName)
                                     then (fetch (NoteFile FullFileName) of NoteFileOrFileName)
                                   elseif NoteFileOrFileName
                                   else (NC.DatabaseFileName "Name of NoteFile to open:" " -- " T NIL NIL
                                                             TempInterestedWindow]
            then (RETURN NIL))

;;; Check for the .NoteFile extension if this notefile has not been noticed by NoteCards.

        (if (AND (NEQ (U-CASE (FILENAMEFIELD FileName 'EXTENSION))
                      'NOTEFILE)
                 (NOT (NC.NoteFileNoticedP FileName)))
            then (NC.PrintMsg InterestedWindow T FileName " does not have a .NOTEFILE extension." (CHARACTER
                                                                                                    13)
                              "Are you sure you want to open " FileName " as a NoteFile?")
                 (if (NULL (NC.AskYesOrNo "" " -- " "No" NIL InterestedWindow T NIL))
                     then (NC.ClearMsg InterestedWindow T)
                          (RETURN NIL)))

    ;; If this is an open NoteFIle, just bring up its menu.

        (if (NC.NoteFileOpenP FileName)
            then (NC.SetUpNoteFileInterface (NC.NoteFileFromFileName FileName)
                                            MenuPosition TempInterestedWindow Don'tCreateInterfaceFlg)
                 (NC.ClearMsg TempInterestedWindow T)          (* bring up or create notefile icon if needed)
                 (RETURN NIL))

    ;; Check for higher version of same notefile

        [if (AND (NOT Don'tCheckVersionFlg)
                 (SETQ OldVersion (FILENAMEFIELD FileName 'VERSION))
                 (SETQ NewVersion (FILENAMEFIELD (SETQ NewerFileName (FULLNAME (PACKFILENAME 'VERSION NIL
                                                                                            'BODY FileName)))
                                                 'VERSION))
                 (LESSP OldVersion NewVersion))
            then ;; Notify user
                 (NC.PrintMsg TempInterestedWindow T "A higher version of " FileName " exists." (CHARACTER
                                                                                                 13)
                              "Open " NewerFileName " instead? (Highest version)")
                 ;; Open the version the user requests.
                 (if (NC.AskYesOrNo "" " -- " "No" NIL TempInterestedWindow T NIL)
                     then (SETQ FileName NewerFileName)
                          (SETQ NoteFileOrFileName (NC.NoteFileFromFileName FileName))
                          (if (SETQ NoteFileMenu (NC.GetNoteFileMenu FileName))
                              then (NC.SetNoteFileMenu NoteFileOrFileName NoteFileMenu]

    ;; Create a NoteFile object or use existing notefile object if there is one for this file name.

        [SETQ NoteFile (if (type? NoteFile NoteFileOrFileName)
                           then NoteFileOrFileName
                         else (OR (NC.NoteFileFromFileName FileName)
                                  (create NoteFile]
        (replace (NoteFile FullFileName) of NoteFile with FileName)
        (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow NoteFile)))
```

```
(RETURN
 (NC.ProtectedNoteFileOperation
  NoteFile "Open NoteFile" InterestedWindow
  (PROG NIL
   ;; Figure out the appropriate device vector from the file name.
        (NC.InstallDeviceVectorInNoteFile NoteFile PublicOrPrivate)
   ;; Moved this replace to after test for open notefile.  Otherwise, if notefile is open read-only, it well be changed to regular open.
        (replace (NoteFile ReadOnlyFlg) of NoteFile with ReadOnlyFlg)
   ;; Notify user.
        (OR QuietFlg (NC.PrintMsg InterestedWindow T "Opening NoteFile: " FileName " ..." (CHARACTER
                                                                                            13)))
        (SETQ ReturnValue
         (PROG NIL
          ;; Run through OpenNoteFileFns with param of BEFORE.  Exit if any returns DON'T
              (if [for Function in NC.OpenNoteFileFns thereis (OR (EQ Function 'DON'T)
                                                                  (EQ 'DON'T (APPLY* Function FileName
                                                                                     NoteFile 'BEFORE
                                                                                     InterestedWindow]
                  then (if (WINDOWP InterestedWindow)
                           then (NC.PrintMsg InterestedWindow NIL "Open canceled for NoteFile "
                                     FileName "." (CHARACTER 13))
                                (DISMISS NC.MsgDelay)
                                (NC.ClearMsg InterestedWindow T))
                       (RETURN))
          ;; Call the device specific OpenNoteFileFn, which returns a list of special UIDs
              (if [NULL (ERSETQ (SETQ ReturnValue (APPLY* (fetch (NoteFile OpenNoteFileFn) of NoteFile)
                                                          NoteFile InterestedWindow
                                                          Don'tCheckForTruncationFlg]
                  then (SETQ ReturnValue 'NoteFileOpenFailed))
          ;; Process error returns from the OpenNoteFileFn
              (if (NOT (LITATOM ReturnValue))
                  then ;; OpenNoteFileFn returned correctly
                       (SETQ CriticalUIDs ReturnValue)
                  else ;; Error, process it.
                       (SETQ ReturnValue
                        (OR [SELECTQ ReturnValue
                                (NoteFileNotFound
                                    (NC.RemoveAccessToNoteFile NoteFile)
                                    (SETQ InterestedWindow (NC.CoerceToInterestedWindow
                                                               InterestedWindow))
                                    (NC.ProcessNoteFileNotFoundError NoteFile Don'tCacheTypesAndTitlesFlg
                                           Don'tCreateFlg ConvertNoConfirmFlg Don'tCreateArrayFlg
                                           Can'tTruncateFlg Don'tCreateInterfaceFlg
                                           Don'tGetSpecialCardsFlg InterestedWindow PublicOrPrivate
                                           MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg
                                           ))
                                (NoteFileNeedsConversion
                                    (NC.ProcessNoteFileNeedsConversionError NoteFile
                                           Don'tCacheTypesAndTitlesFlg Don'tCreateFlg
                                           ConvertNoConfirmFlg Don'tCreateArrayFlg Can'tTruncateFlg
                                           Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg
                                           InterestedWindow PublicOrPrivate MenuPosition QuietFlg
                                           ReadOnlyFlg Don'tCheckForTruncationFlg))
                                (NoteFileNeedsTruncation
                                    (NC.ProcessNoteFileNeedsTruncationError NoteFile
                                           Don'tCacheTypesAndTitlesFlg Don'tCreateFlg
                                           ConvertNoConfirmFlg Don'tCreateArrayFlg Can'tTruncateFlg
                                           Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg
                                           InterestedWindow PublicOrPrivate MenuPosition QuietFlg
                                           ReadOnlyFlg Don'tCheckForTruncationFlg))
                                (NoteFileAlreadyOpen
                                    (ERSETQ (NC.ReportError NIL (CONCAT (fetch (NoteFile FullFileName)
                                                                           of NoteFile)
                                                                        " is already open for exclusive
                                                                        access. Open failed."))))
                                (NoteFileOpenFailed
                                    (ERSETQ (NC.ReportError NIL (CONCAT "Open of " (fetch (NoteFile
                                                                                          FullFileName)
                                                                                     of NoteFile)
                                                                        " failed for unknown reason.")))
)
                                ((NoteFileHeaderBad BadNextIndexNum BadHashArraySize BadCheckptPtr
                                       BadNextLinkNum)
                                    (ERSETQ (NC.ReportError NIL (CONCAT "Header of NoteFile "
                                                                        (fetch (NoteFile FullFileName)
                                                                           of NoteFile)
                                                                        " is bad: " ReturnValue ".
                                                                        Contact a NoteCards wizard."))))
                                (PROGN (ERSETQ (NC.ReportError NIL (CONCAT "Unknown error code ("
```

```
                                                                    ReturnValue ") returned by
                                                                    OpenNoteFileFn for NoteFile "
                                                                    FileName]
                    ReturnValue))

          ;; notify the user.  if there's been a problem

          (if (AND (NOT (type? NoteFile ReturnValue))
                   (WINDOWP InterestedWindow))
              then (NC.PrintMsg InterestedWindow NIL "Open canceled for NoteFile " FileName
                          "." (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg InterestedWindow T))

          ;; return whatever the error processing returned.

          (RETURN ReturnValue))
     (SETQ ReturnValue)

;; Make sure there is no other open NF with this UID.

     (LET (NF)
          (if (AND (SETQ NF (GETHASH (fetch (NoteFileCriticalUIDs NoteFile) of CriticalUIDs)
                                     NC.NoteFilesHashArray))
                   (NEQ (fetch (NoteFile FullFileName) of NoteFile)
                        (fetch (NoteFile FullFileName) of NF))
                   (NC.NoteFileOpenP NF))
              then (FLASHW PROMPTWINDOW)
                   (NC.PrintMsg PROMPTWINDOW T "Couldn't open " FileName (CHARACTER 13)
                          "because "
                          (fetch (NoteFile FullFileName) of NF)
                          " is already open "
                          (CHARACTER 13)
                          "and has the same UID.")
                   (NC.CloseNoteFile NoteFile InterestedWindow T)
                   (RETURN NIL)))

;; If needed, build a hash array by calling the device specific BuilHashArrayFn.

     (replace (NoteFile UID) of NoteFile with (fetch (NoteFileCriticalUIDs NoteFile)
                                                     of CriticalUIDs))

;; Store this NoteFile object in the appropriate NoteFile hash array

     (NC.NoticeNoteFile NoteFile)
     (if (NOT Don'tCreateArrayFlg)
         then (OR QuietFlg (NC.PrintMsg InterestedWindow T "Opening NoteFile: " FileName
                                    (CHARACTER 13)
                                    "Building index array ..."
                                    (CHARACTER 13)))
              (if (OR [NULL (ERSETQ (SETQ ReturnValue (APPLY* (fetch (NoteFile
                                                                     BuildHashArrayFn)
                                                                 of NoteFile)
                                                              NoteFile QuietFlg
                                                              InterestedWindow
                                                              (CONCAT "Opening NoteFile "
                                                                      (fetch (NoteFile
                                                                                 FullFileName
                                                                                 )
                                                                             of NoteFile)
                                                                      (CHARACTER 13]
                      (NOT (type? NoteFile ReturnValue)))
                  then ;; Error during building of hash array
                       (ERSETQ (NC.ReportError NIL (CONCAT "Build Hash Array failed for
                                                           NoteFile " (fetch (NoteFile
                                                                                 FullFileName
                                                                                 )
                                                                             of NoteFile)
                                                           " because " ReturnValue)))
                       (RETURN)))

;; Set up critical UIDs in NoteFile object using the values returned from OpenNoteFileFn.

     (NC.InstallCriticalUIDsInNoteFile NoteFile CriticalUIDs)

;; if needed, cache the special cards

     [if (NOT Don'tGetSpecialCardsFlg)
         then (NC.GetSpecialCards NoteFile QuietFlg InterestedWindow
                    (CONCAT "Opening NoteFile: " (fetch (NoteFile FullFileName) of NoteFile)
                            (CHARACTER 13]

;; If needed, start the titles and types caching process

     [if (NOT Don'tCacheTypesAndTitlesFlg)
         then (replace (NoteFile CachingProcess) of NoteFile
                 with (ADD.PROCESS (LIST (FUNCTION NC.CacheTypesAndTitles)
                                         NoteFile]

;; If needed, open up a NoteFile interface.

     (NC.SetUpNoteFileInterface NoteFile MenuPosition InterestedWindow
            Don'tCreateInterfaceFlg)

;; Record this as the last NF opened.
```

```
                    (SETQ NC.LastNoteFileOpened NoteFile)
                    (RETURN NoteFile)))
               (if (type? NoteFile ReturnValue)
                   then ;; Run through OpenNoteFIleFns with param of AFTER.  Stop if any returns DON'T
                        (for Function in NC.OpenNoteFileFns thereis (EQ 'DON'T (APPLY* Function FileName
                                                                                       NoteFile 'AFTER
                                                                                       InterestedWindow)))
                        ;; Go home, returning NoteFile
                        (if (NULL QuietFlg)
                            then (NC.PrintMsg InterestedWindow T "Opening NoteFile: " FileName (CHARACTER
                                                                                                13)
                                              "Done."
                                              (CHARACTER 13))
                                 (NC.ClearMsg InterestedWindow T))
                        ;; Bring up the TOC if requested
                        [if BringUpTOC
                            then (for CARD in (NC.FetchTopLevelCards NoteFile) when (EQUAL (NC.RetrieveTitle
                                                                                            CARD)
                                                                                           "Table of Contents")
                                    do (NC.EditNoteCard CARD (fetch (NoteFile ReadOnlyFlg) of NoteFile]
                        (RETURN NoteFile)
                   else ;; Bail out if open was unsuccessful.
                        (RETURN NIL])
```

(NC.CacheTypesAndTitles

```
[LAMBDA (NoteFile)                                          (* fgh%: " 3-Sep-86 19:51")
```

(* Cache or uncache all of the titles on DatabaseSteam onto the prop lists of the NoteCard IDs)

(* * rht 7/27/85%: Added BLOCK call to cut down CPU hogging.)

(* * kirk |10/18/85| changed printmessage from every 10 to every 50.0 Replaced call on NC.GetTypeAndTitle with direct calls on NC.RetrieveType and NC.RetrieveTitle.)

(* * fkr 10/29/85%: Added support for numeric IDs and cache arrays.)

(* * rht 11/13/85%: Updated to handle new notefile and Card formats.)

(* * fgh |9/3/86| Adapted to use NC.RetrieveTypeAndTitle to speed up caching on server.)

```
   (NC.MapCards NoteFile (FUNCTION (LAMBDA (Card)
                                   (NC.RetrieveTypeAndTitle Card])
```

(NC.AskUserAboutTruncation

```
[LAMBDA (NoteFile PromptWindow PromptMsg)                   ; Edited  8-Dec-88 11:09 by krivacic
```

;;; Ask the user what to do since NoteFile truncation is needed.

;;; fgh 5/25/86 First created.

;;; rht 10/29/86: Change name from Abort to Cancel.

;;; pmi 3/25/87: Added NC.MenuFont to all menus

;;; pmi 6/25/87: Replaced menu with NC.AskUserWithMenu.

;;; pmi 7/1/87: Added PromptMsg argument.

```
   (DECLARE (GLOBALVARS NC.MenuFont))
   (LET (Items Menu PromptWinPos)
        ;; List user's options.  Include Inspect&Repair only if this NoteFIle device has a RepairFn.
        [SETQ Items `((Cancel 'Cancel "Cancel the Open.")
                      (Truncate% File 'Truncate% File "Truncate the file throwing away changes since last
                                checkpoint.")
                      ,@(if (fetch (NoteFile RepairNoteFileFn) of NoteFile)
                            then (LIST '(|Inspect & Repair| '|Inspect & Repair| "Run Inspect&Repair on this
                                                NoteFile."]
        ;; Bring up the menu with the options.
        (CADR (CADR (NC.AskUserWithMenu Items PromptMsg PromptWindow])
```

(NC.InstallCriticalUIDsInNoteFile

```
[LAMBDA (NoteFile CriticalUIDs)                             (* fgh%: "24-May-86 12:23")
```

(* * Install the contents of the critical UIDs record into the NoteFile object.)

```
   (LET [(HashArray (HARRAYP (fetch (NoteFile HashArray) of NoteFile]
        (if (NOT (NC.SameUIDP (fetch (NoteFile UID) of NoteFile)
                              (fetch (NoteFileCriticalUIDs NoteFile) of CriticalUIDs)))
```

```
            then (ERSETQ (NC.ReportError NIL "Mismatch in NoteFile UIDs."))
          else [replace (NoteFile TableOfContentsCard) of NoteFile with (if HashArray
                                                                            then (NC.CardFromUID
                                                                                   (fetch (NoteFileCriticalUIDs
                                                                                            TableOfContents)
                                                                                     of CriticalUIDs)
                                                                                   NoteFile)
                                                                          else (create Card
                                                                                       UID _ (fetch (
                                                                                                  NoteFileCriticalUIDs
                                                                                                    TableOfContents)
                                                                                               of CriticalUIDs]
               [replace (NoteFile OrphansCard) of NoteFile with (if HashArray
                                                                    then (NC.CardFromUID (fetch (
                                                                                                 NoteFileCriticalUIDs
                                                                                                  Orphans)
                                                                                           of CriticalUIDs)
                                                                           NoteFile)
                                                                  else (create Card
                                                                               UID _ (fetch (NoteFileCriticalUIDs
                                                                                             Orphans)
                                                                                       of CriticalUIDs]
               [replace (NoteFile ToBeFiledCard) of NoteFile with (if HashArray
                                                                      then (NC.CardFromUID (fetch (
                                                                                                   NoteFileCriticalUIDs
                                                                                                     ToBeFiled)
                                                                                             of CriticalUIDs)
                                                                             NoteFile)
                                                                    else (create Card
                                                                                 UID _ (fetch (NoteFileCriticalUIDs
                                                                                               ToBeFiled)
                                                                                         of CriticalUIDs]
               [replace (NoteFile LinkLabelsCard) of NoteFile with (if HashArray
                                                                       then (NC.CardFromUID (fetch (
                                                                                                    NoteFileCriticalUIDs
                                                                                                      LinkLabels)
                                                                                              of CriticalUIDs)
                                                                              NoteFile)
                                                                     else (create Card
                                                                                  UID _ (fetch (NoteFileCriticalUIDs
                                                                                                LinkLabels)
                                                                                          of CriticalUIDs]
               (replace (NoteFile RegistryCard) of NoteFile with (if HashArray
                                                                     then (NC.CardFromUID (fetch (
                                                                                                  NoteFileCriticalUIDs
                                                                                                    Registry)
                                                                                            of CriticalUIDs)
                                                                            NoteFile)
                                                                   else (create Card
                                                                                UID _ (fetch (NoteFileCriticalUIDs
                                                                                              Registry)
                                                                                        of CriticalUIDs])
```

(NC.ProcessInspectAndRepairRequest

```
  [LAMBDA (NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg Don'tCreateArrayFlg
                Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg PromptWindow PublicOrPrivate
                MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg)
                                                                    (* pmi%: "30-Jun-87 17:04")
```

(* * Do an an InspectAndRepair on a file then try to reopen. Used when checkpoint pinter out of sorts during an Open.)

(* * fgh |5/25/86| First created.)

(* * fgh |9/1/86| Updated args to match NC.OpenNoteFile.)

(* * rht 10/31/86%: Changed outdated call to NC.RepairNoteFile to NC.InspectAndRepairNoteFile.
Added Don'tCheckForTruncationFlg arg. No longer tries to open after NC.InspectAndRepairNoteFile returns.)

(* * pmi 5/14/87%: Removed NoteFilesHashArray argument.)

(* * pmi 6/26/87%: Once again tries to open after NC.InspectAndRepairNoteFile returns.)

(* * pmi 6/30/87%: Removed extra Don'tCheckOperationsInProgressFlg argument.
Added check for a NoteFile object returned from NC.InspectAndRepairNoteFile.)

```
    (if [type? NoteFile (CAR (ERSETQ (NC.InspectAndRepairNoteFile NoteFile NIL PromptWindow]
        then
```

(* * Repair NoteFile finished, try to Open the NF now.)

```
          (NC.OpenNoteFile NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg
                 Don'tCreateArrayFlg Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg
                 PromptWindow PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg)
      else
```

(* * Repair NoteFile failed for some reason. Report and return.)

```
        [ERSETQ (NC.ReportError NIL (CONCAT "NoteFile Inspect&RepairFailed for NoteFile " (fetch (NoteFile
                                                                                                  FullFileName)
                                                                                             of NoteFile]
        'InspectAndRepairFailed])
```

(NC.ProcessNoteFileNotFoundError

```
[LAMBDA (NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg Don'tCreateArrayFlg
               Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg InterestedWindow
               PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg)
                                                                    ; Edited 19-Oct-88 10:17 by RAR
```

(* * NoteFile couldnot be found when an attempt was made to open it.
Find out if the user wants to create it.)

(* * fgh |5/23/86| First created.)

(* * fgh |9/1/86| Updated args to match NC.OpenNoteFIle.)

(* * rht 10/31/86%: Added Don'tCheckForTruncationFlg arg.)

(* * pmi 5/28/87%: Removed NoteFilesHashArray argument.)

(* * dsj. |11/5/87.| Now correctly passes a multitude of args to NC.CreateNoteFile.)

;; rar: 10/19/88 Added check so that we can create the NF silently when it doesn't exist.

```
(LET (NewFileName)
```

(* * Notify user of the error if we're not gagged)

```
(OR QuietFlg (NC.PrintMsg InterestedWindow T "Could not find NoteFile " (fetch (NoteFile FullFileName)
                                                                            of NoteFile)
                    (CHARACTER 13)))
```

(* * If the user wants to create the file, then create it and try the open again.)

;; If the quiet flg is true, then go ahead and create the notefile without asking for confirmation.

```
(if (AND (NULL Don'tCreateFlg)
         (OR QuietFlg (NC.AskYesOrNo (CONCAT "Do you want to create " (fetch (NoteFile FullFileName)
                                                                        of NoteFile))
                           " -- " "Y" NIL InterestedWindow T NIL)))
    then (if [CAR (ERSETQ (SETQ NewFileName (NC.CreateNoteFile NoteFile NIL NIL InterestedWindow
                                                (CONCAT "Opening NoteFile" (CHARACTER 13))
                                                QuietFlg PublicOrPrivate NIL ReadOnlyFlg
                                                Don'tCreateInterfaceFlg MenuPosition]
             then (NC.OpenNoteFile NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg
                        Convertw/oConfirmFlg Don'tCreateArrayFlg Can'tTruncateFlg
                        Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg InterestedWindow
                        PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg))
     else 'NoteFileNotFound])
```

(NC.ProcessTruncationRequest

```
[LAMBDA (NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg Don'tCreateArrayFlg
               Don'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg PromptWindow PublicOrPrivate
               MenuPosition QuietFlg ReadOnlyFlg)              (* pmi%: "14-May-87 11:42")
```

(* * Do the actual truncation of a NoteFile that has stuf past the checkpoint pointer.)

(* * fgh |5/25/86| First created.)

(* * fgh |9/1/86| Updated args to match NC.OpenNoteFile.)

(* * rht 2/19/87%: Now passes PromptWindow arg to TruncationFn.)

(* * pmi 5/14/87%: Removed NoteFilesHashArray argument.)

```
(LET (ReturnValue (TruncationFn (fetch (NoteFile TruncateNoteFileFn) of NoteFile)))
```

(* * Do the truncation.)

```
(if [type? NoteFile (CAR (ERSETQ (SETQ ReturnValue (APPLY* TruncationFn NoteFile PromptWindow]
    then
```

(* * Truncation successful, Open the note file.)

```
        (NC.OpenNoteFile NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg
              Don'tCreateArrayFlg Don'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg
              PromptWindow PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg)
 else
```

(* * Error during truncation fn. report and get out of here.)

```
     (SELECTQ ReturnValue
         (NoteFileTruncationAborted
             'NoteFileTruncationAborted)
```

```
                (PROGN (ERSETQ (NC.ReportError NIL (CONCAT "NoteFile truncation failed for NoteFile "
                                                           (fetch (NoteFile FullFileName) of NoteFile)
                                                           " because " ReturnValue ".")))
                       'NoteFileTruncationFailed])
```

(NC.ProcessNoteFileNeedsConversionError

```
[LAMBDA (NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg Don'tCreateArrayFlg
          Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg PromptWindow PublicOrPrivate
          MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg)
                                                                    (* pmi%: "14-May-87 11:38")
```

(* * Discovered NoteFile needs version conversion when attempted to open it.
Process that error.)

(* * fgh |5/23/86| First created.)

(* * fgh |9/1/86| Updated args to match NC.OpenNoteFile.)

(* * rht 10/31/86%: Added Don'tCheckForTruncationFlg arg.)

(* * 11/6/86%: Now passes PromptWindow down to conversion fn.)

(* * pmi 5/14/87%: Removed NoteFilesHashArray argument.)

```
  (LET (ConversionFn ReturnValue)
```

(* * If appropriate, notify the user.)

```
       (if (WINDOWP PromptWindow)
           then (NC.PrintMsg PromptWindow T "NoteFile " (fetch (NoteFile FullFileName) of NoteFile)
                      "is an old format file."
                      (CHARACTER 13)))
```

(* * If the device has a ConversionFn, then apply it and try to Open the NoteFile.)

```
       (if (SETQ ConversionFn (fetch (NoteFile ConvertNoteFileFormatFn) of NoteFile))
           then
```

(* * ask the user if conversion should be done. If so, do it, else return an error msg,)

```
               (if (OR Convertw/oConfirmFlg (NC.AskYesOrNo "Do you want to convert it to the new format? "
                                                  " -- " "Y" NIL PromptWindow))
                   then
```

(* * go ahead and convert.)

```
                       [if [type? NoteFile (CAR (ERSETQ (SETQ ReturnValue (APPLY* ConversionFn NoteFile
                                                                                 PromptWindow]
                           then
```

(* * Conversion successful, Open the note file.)

```
                                   (NC.OpenNoteFile NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg
                                          Convertw/oConfirmFlg Don'tCreateArrayFlg Can'tTruncateFlg
                                          Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg PromptWindow
                                          PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg
                                          Don'tCheckForTruncationFlg)
                               else
```

(* * Error during conversion fn. report and get out of here.)

```
                           (SELECTQ ReturnValue
                               (PROGN [ERSETQ (NC.ReportError NIL (CONCAT "NoteFile format conversion
                                                                         failed for NoteFile "
                                                                         (fetch (NoteFile FullFileName)
                                                                            of NoteFile]
                                              'NoteFileConversionFailed]
                       else
```

(* * User said don't convert. Just return)

```
                   'NoteFileNeedsConversion)
        else
```

(* * No conversion fn. Tell the user if appropriate and return an error msg.)

```
             (if (WINDOWP PromptWindow)
                 then (NC.PrintMsg PromptWindow NIL "No format conversion possible." (CHARACTER 13)
                            "See a NoteCards wizard."
                            (CHARACTER 13)))
             'NoteFileNeedsConversion])
```

(NC.ProcessNoteFileNeedsTruncationError

```
[LAMBDA (NoteFile Don'tCacheTypesAndTitlesFlg Don'tCreateFlg Convertw/oConfirmFlg Don'tCreateArrayFlg
          Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg PromptWindow PublicOrPrivate
```

```
                MenuPosition QuietFlg ReadOnlyFlg Don'tCheckForTruncationFlg)
                                                                    ; Edited  8-Dec-88 11:09 by krivacic
```

;;; Discovered NoteFile needs version truncation when attempted to open it. Process that error.

;;; fgh 5/23/86 First created.

;;; fgh 9/1/86 Updated args to match NC.OpenNoteFile.

;;; rht 10/29/86: Changed name from Abort to Cancel.

;;; rht 10/31/86: Added Don'tCheckForTruncationFlg arg.

;;; pmi 5/14/87: Removed NoteFilesHashArray argument.

;;; pmi 6/2/87: Reinstates fix made by Randy T. which I must have clobbered with the above patch file of my own: (* * rht 4/30/87: No longer breaks ;;; when user selects outside of menu.)

;;; pmi 7/1/87: Added PromptMsg argument to NC.AskUserAboutTruncation.

```
    (PROG (TruncationFn)
     ;; If appropriate, notify the user.
          (if (WINDOWP PromptWindow)
              then (NC.PrintMsg PromptWindow T "NoteFile " (fetch (NoteFile FullFileName) of NoteFile)
                         " has information written since last successful close or checkpoint."
                         (CHARACTER 13)))
     ;; If the device has a TruncationFn then apply it and try to Open the NoteFile.
          (if (AND (NULL Can'tTruncateFlg)
                   (SETQ TruncationFn (fetch (NoteFile TruncateNoteFileFn) of NoteFile)))
              then ;; ASk the user if they want to truncate, or abort, or repair the notefile.
                   (SELECTQ (NC.AskUserAboutTruncation NoteFile PromptWindow (CONCAT "NoteFile "
                                                                                     (fetch (NoteFile FullFileName)
                                                                                        of NoteFile)
                                                                                     " has information written
                                                                                     since last successful close or
                                                                                     checkpoint."
                                                                                     (CHARACTER 13)))
                       (Cancel (RETURN 'CancelOpen))
                       (|Inspect & Repair|
                            (RETURN (NC.ProcessInspectAndRepairRequest NoteFile Don'tCacheTypesAndTitlesFlg
                                          Don'tCreateFlg Convertw/oConfirmFlg Don'tCreateArrayFlg
                                          Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg
                                          PromptWindow PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg
                                          Don'tCheckForTruncationFlg)))
                       (Truncate% File
                            (RETURN (NC.ProcessTruncationRequest NoteFile Don'tCacheTypesAndTitlesFlg
                                          Don'tCreateFlg Convertw/oConfirmFlg Don'tCreateArrayFlg
                                          Can'tTruncateFlg Don'tCreateInterfaceFlg Don'tGetSpecialCardsFlg
                                          PromptWindow PublicOrPrivate MenuPosition QuietFlg ReadOnlyFlg
                                          Don'tCheckForTruncationFlg)))
                       (RETURN 'CancelOpen))
            else
          (* * No TruncationFn. Tell the user if appropriate and return an error msg.)
                (if (WINDOWP PromptWindow)
                    then (NC.PrintMsg PromptWindow NIL "No file truncation possible for this NoteFile."
                                (CHARACTER 13)
                                "See a NoteCards wizard."
                                (CHARACTER 13)))
                (RETURN 'NoteFileNeedsTruncation])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.OpenNoteFileFns NC.CloseNoteFileFns)
)

(RPAQ? NC.OpenNoteFileFns NIL)

(RPAQ? NC.CloseNoteFileFns NIL)
```

;;; Open events card

```
(DEFINEQ
```

(NC.RunOpenEvents

```
  [LAMBDA (FileName NoteFile When InterestedWindow)            (* rht%: " 2-Apr-87 14:26")
```

(* * If When = AFTER then read and eval the OpenEvents card.)

(* * rht 4/2/87%: Added InterestedWindow arg.)

```
    (SELECTQ When
        (AFTER (OR (type? NoteFile NoteFile)
                   (SETQ NoteFile (NC.NoteFileFromFileName FileName)))
               (LET ((OpenEventsCard (NCP.LookupCardByName 'OpenEventsCard NoteFile)))
                    (if (NCP.ValidCardP OpenEventsCard)
                        then (if (NOT (NCP.ActiveCardP OpenEventsCard))
                                 then (NCP.CacheCards OpenEventsCard))
                             (NC.EvaluateListCardSubstance OpenEventsCard "Evaluating the Open Events card"
                                    InterestedWindow))))
        NIL])
```

(NC.RunCloseEvents

```
  [LAMBDA (NoteFileOrName When)                                    (* pmi%: " 5-Jan-88 14:49")

          (* * If When = AFTER then read and eval the CloseEvents card.)

          (* * dsj 9/22/87%: Dropped "NoteFileName" as the first arg passed in and changed to just NoteFileOrName.)

          (* * dsj 9/23/87%: BEFORE/AFTER needed major rehacking%: BEFORE now means before the NF is closed
          (i.e., the NF stream still exists) and AFTER means after the stream has been closed.
          Because we can't retrieve the registry card in the AFTER case, need to save its content special during BEFORE execution.
          "When" can be referenced by the called closing fns.)

          (* * pmi 1/5/88%: added dsj's changes; see above comments.)

    (LET (NoteFile)
         (SETQ NoteFile (if (type? NoteFile NoteFileOrName)
                            then NoteFileOrName
                          else (NC.NoteFileFromFileName NoteFileOrName)))
         (SELECTQ (U-CASE When)
             (BEFORE (LET ((CloseEventsCard (NCP.LookupCardByName 'CloseEventsCard NoteFile)))
                          (if (NCP.ValidCardP CloseEventsCard)
                              then

          (* * dsj%: Save CloseEventsCard substance for AFTER execution.)

                                   (PUTPROP (MKATOM NoteFile)
                                          'CloseEvents
                                          (NCP.CardSubstance CloseEventsCard))
                                   (NC.EvaluateListCardSubstance CloseEventsCard "Evaluating the CloseEvents
                                          card before" (NCP.NoteFileIconWindow NoteFile)
                                          T))))
             (AFTER (LET [(CloseEvents (GETPROP (MKATOM NoteFile)
                                                'CloseEvents]
                         (if CloseEvents
                             then (NC.EvaluateListCardSubstance CloseEvents "Evaluating the Close Events card
                                         after" (NCP.NoteFileIconWindow NoteFile)
                                         T))))
             NIL])

)

(ADDTOVAR NC.OpenNoteFileFns NC.RunOpenEvents)

(ADDTOVAR NC.CloseNoteFileFns NC.RunCloseEvents)
```

;;; closing notefiles.

```
(DEFINEQ
```

(NC.CloseDatabaseFile

```
  [LAMBDA (NoteFile InterestedWindow)                              (* Randy.Gobbel " 5-Nov-86 16:10")
    (NC.CloseNoteFile NoteFile])
```

(NC.CloseNoteFile

```
  [LAMBDA (NoteFile InterestedWindow QuietFlg AutoConfirmFlg)      ; Edited  3-Dec-87 18:59 by rht:

          (* * Close a NoteFIle)

          (* * rht 10/23/84%: Now gives user option of closing and saving all open cards on the screen.)

          (* * rht 11/8/84%: Put RESETLST around NC.CacheTitles call.)

          (* * rht 1/9/85%: Clear the NC.UncachingNotCompleted variable when close successfully completes.)

          (* * rht 1/31/85%: Added call to checkpoint database. That in turn dumps the next nodeID and next linkID.)

          (* * rht 7/14/85%: Replaced the call to reset the main menu with call to NC.ResetMainMenu.
          Also took out redundant reset of PSA.Database, since NC.ForceDatabaseClose is doing that.)

          (* * fgh |10/16/85| removed call to CacheTypesAndTitles because uncacheing now done automatically by cache
          mechanism.)
```

(* * fkr 10/29/85%: Now kills caching process from database streamprop.)

(* * fkr |11/8/85| Updated to handle new NoteFile object and new CardID scheme.)

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

(* * rht 3/26/86%: Now searches for active cards over whole notefile not just among cards up on screen.
Uses NC.MapCards.)

(* * kirk 28Apr86 Now returns NoteFile if successful.)

(* * fgh |5/2/86| Cleaned up. Ask user to confirm only if there are cards on the screen, not if there are active, but not displayed ones. Added calls to the NC.CloseNoteFileFns before and after the closeing.)

(* * fgh |5/26/86| Revamp for device vector implementation.)

(* * kef 7/24/86%: Changed the last expression at the end that smashes the NoteFile device out of the NoteFile data structure. This is so that the Interface will not bomb trying to apply an OPENP function with a NIL Device.)

(* * kef 8/4/86%: Added something to obtain the write lock on the parts of the active NoteCards that deactivating will release. This is also so that any changes may be written to the server.)

(* * fgh |8/31/86| Reimplemented changes in system made since |5/23/86| conversion.
Reimplemented changes include%: (* * fgh |6/4/86| Fixed so that shrunken cards are counted as open when asking for confirmation when there are open cards on screen.) (* * fgh |6/13/86| Now checks for card operations in progress and kills them if necessary.) (* * fgh |6/25/86| Added NC.ProtectedNoteFileOperation macro call.
Added Don't CheckOperationInProgressFlg args.) (* * rht 7/4/86%: Added check for readonly notefile.)
(* * rht 7/13/86%: Added QuietFlg arg. Note that this will cause open cards on the screen to be closed and saved without asking user for confirmation.) (* * rpr 11/13/86%: After closing active cards, checks to see if any special cards were made active and closes them.))

(* * pmi |12/22/86| Made test for open notefile consistent with other NoteFile operations
(ie. Abort Checkpoint,))

(* * rht 2/16/87%: Added AutoConfirmFlg argument to prevent user having to confirm whether to close and save open cards. Note that QuietFlg is stronger than AutoConfirmFlg in that other messages are suppressed as well.)

(* * rg |3/4/87| changes for new concurrency machinery)

(* * rht 3/25/87%: Now calls NC.CoerceToInterestedWindow.)

(* * rg |3/27/87| redid concurrency wrapper)

(* * pmi 8/14/87%: Added call to NC.NoticeNoteFile to make sure this file has been noticed.
Also, added parameters in call to NC.AbortSession to stop confirmation and pass on the QuietFlg.)

```
(DECLARE (GLOBALVARS NC.MsgDelay NC.CloseNoteFileFns))
(NC.ProtectedNoteFileOperation
 NoteFile "Close NoteFile" InterestedWindow (OR (OPENWP InterestedWindow)
                                                (SETQ InterestedWindow (NC.CoerceToInterestedWindow NoteFile
                                                                                                    )))
 (if (NC.ReadOnlyNoteFileP NoteFile)
     then (NC.AbortSession NoteFile InterestedWindow T QuietFlg)
   else
   (ALLOW.BUTTON.EVENTS)
   (PROG ((FullFileName (fetch (NoteFile FullFileName) of NoteFile))
          CardTotal ActiveCards ReturnValue (OperationMsg ""))

      (* * Make sure NF is open)

         (if [NULL (ERSETQ (SETQ ReturnValue (NC.NoteFileOpenP NoteFile]
             then (SETQ ReturnValue 'OpenPFailed))
         (if (NULL ReturnValue)
             then

      (* * NoteFile is not open.)

                 (NC.PrintMsg InterestedWindow T "Can't close a closed notefile." (CHARACTER 13))
                 (DISMISS NC.MsgDelay)
                 (NC.ClearMsg InterestedWindow T)
                 (RETURN NIL)
           elseif (NOT (type? NoteFile ReturnValue))
             then

      (* * Error return from NoteFileOpenPFn)

                 (if [NULL (ERSETQ (NC.ReportError NIL (CONCAT "OpenP test on " FullFileName "failed because
" ReturnValue (CHARACTER 13)
                                                               "OK to continue Close. ^ to abort Close."]
                     then (RETURN ReturnValue)))
         (RETURN
          (PROG NIL
                (RESETSAVE NIL `(NC.ClearMsg ,InterestedWindow T))

      (* * Delete the types and titles caching process if still alive. Have to do it now in order to make checking operations that
      follow suitably efficient. Note its a bit too early since we can still cancel this close.
      But any harm done is loss of speed if NoteFile remains open when close iss cancelled.)
```

```
                (DEL.PROCESS (fetch (NoteFile CachingProcess) of NoteFile))
```

(* * See if any cards have operations in progress. If so, kill them after confirming with user.)

```
                (OR QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                    "Checking for card operations in progress ..."
                                    (CHARACTER 13)))
                (if (EQ 'ABORT (NC.CardOperationsInProgress NoteFile T))
                    then (RETURN NIL))
                (NC.ClearMsg InterestedWindow NIL)
```

(* * If NULL QuietFlg then look for cards on the screen. If there are active cards ask the user if they still want to close. When there's a non-NIL QuietFlg we just close the active cards.)

```
                (if (AND (NULL QuietFlg)
                         (NULL AutoConfirmFlg)
                         [for Window in (OPENWINDOWS)
                            thereis (LET (Card)
                                         (AND [SETQ Card (OR (NC.CardFromWindow Window)
                                                             (AND (WINDOWP (WINDOWPROP Window 'ICONFOR))
                                                                  (NC.CardFromWindow (WINDOWPROP Window
                                                                                                 'ICONFOR]
                                              (NC.SameNoteFileP NoteFile (fetch (Card NoteFile) of Card]
                         (NULL (NC.AskYesOrNo (CONCAT "There are still cards on the screen from this NoteFile
" FullFileName "." (CHARACTER 13)
                                                      "Want to close and save them? ")
                                              " -- "
                                              'Yes NIL InterestedWindow NIL NIL)))
                    then (RETURN NIL))
```

(* * Run through CloseNoteFileFns with param of BEFORE. Exit if any returns DON'T)

```
                (if [for Function in NC.CloseNoteFileFns thereis (OR (EQ Function 'DON'T)
                                                                     (EQ 'DON'T (APPLY* Function NoteFile
                                                                                        'BEFORE]
                    then (RETURN NIL))
```

(* * Close all the active cards)

```
                (OR QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                    "Checking for active cards ..."
                                    (CHARACTER 13)))
                [if (SETQ ActiveCards (NC.MapCards NoteFile [FUNCTION (LAMBDA (Card)
                                                                      Card]
                                                   (FUNCTION NC.ActiveCardP)))
                    then (if (NULL QuietFlg)
                             then (NC.PrintMsg InterestedWindow T "Closing and saving active cards ... "))
                         (RESETLST
                             (RESETSAVE NC.ForceSourcesFlg NIL)
                             (RESETSAVE NC.ForceFilingFlg NIL)
                             (RESETSAVE NC.ForceTitlesFlg NIL)
                             (NC.CloseListOfActiveCards ActiveCards InterestedWindow QuietFlg)
                             (NC.CloseListOfActiveCards (for Card in (NC.FetchSpecialCards NoteFile)
                                                           when (NC.ActiveCardP Card) collect Card)
                                    InterestedWindow QuietFlg))
                         (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "Done." (CHARACTER 13]
```

(* * Checkpoint the NoteFile.)

```
                (OR QuietFlg (NC.PrintMsg InterestedWindow T "Closing Notefile ... " (CHARACTER 13)))
                (if [NULL (CAR (ERSETQ (SETQ ReturnValue (NC.CheckpointNoteFile NoteFile QuietFlg T
                                                                    InterestedWindow OperationMsg]
                    then (SETQ ReturnValue 'CheckpointFailed))
```

(* * Process error returns from in NC.CheckpointNoteFile)

```
                (if (NOT (type? NoteFile ReturnValue))
                    then (if [NULL (ERSETQ (NC.ReportError NIL (CONCAT "Checkpoint of NoteFile " FullFileName
                                                                       " failed because " ReturnValue "."
                                                                       (CHARACTER 13)
                                                                       "OK to continue Close. ^ to abort
                                                                       Close."]
                             then (RETURN ReturnValue)))
```

(* * Close the file.)

```
                (if [NULL (CAR (ERSETQ (SETQ ReturnValue (APPLY* (fetch (NoteFile CloseNoteFileFn)
                                                                    of NoteFile)
                                                                 NoteFile InterestedWindow]
                    then (SETQ ReturnValue 'CloseFailed))
```

(* * Process error returns from the close.)

```
                [if (NOT (type? NoteFile ReturnValue))
                    then (SELECTQ ReturnValue
                                  (NoteFileNotOpen
```

```
                                    (if [NULL (ERSETQ (NC.ReportError NIL (CONCAT "NoteFile" FullFileName " is
                                                                                  not open." (CHARACTER 13)
                                                                                  "OK to continue Close. ^ to
                                                                                  abort Close."]
                                              then (RETURN ReturnValue)))
                                        (PROGN [ERSETQ (NC.ReportError NIL (CONCAT "Close of NoteFile " FullFileName
                                                                                   " failed because " ReturnValue "."
                                                                                   (CHARACTER 13]
                                               (RETURN ReturnValue]
```

(* * Run through CloseNoteFileFns with param of AFTER. Stop if any returns DON'T)

```
            [for Function in NC.CloseNoteFileFns thereis (EQ 'DON'T (APPLY* Function NoteFile 'AFTER]
```

(* * Reset the interface, make sure the notefile has been noticed, and notify the user.)

```
            (NC.ResetNoteFileInterface NoteFile)
            (NC.NoticeNoteFile NoteFile)
            (OR QuietFlg (NC.PrintMsg InterestedWindow T FullFileName " closed."))
```

(* * * Cleanup a bit.)

```
                                                    (* Clean off the card cache's)
            (ADD.PROCESS (LIST (FUNCTION NC.CleanupCardObjects)
                               (fetch (NoteFile HashArray) of NoteFile)))
                                                    (* Clean off the NoteFile object to remove any circularities.)
            (create NoteFile smashing NoteFile Stream _ NIL UID _ (fetch (NoteFile UID) of NoteFile)
                                      FullFileName _ FullFileName Menu _ (fetch (NoteFile Menu) of NoteFile)
                                      NoteFileDevice _ (fetch (NoteFile NoteFileDevice) of NoteFile))
```

(* * Return the NF)

```
            (RETURN NoteFile])
```

(NC.CloseListOfActiveCards

```
  [LAMBDA (ActiveCards InterestedWindow QuietFlg)                    (* rht%: "30-Mar-87 14:49")
```

(* * Close a list of active cards.)

(* * rg |3/4/87| added NC.ProtectedSessionOperation wrapper)

(* * rg |3/12/87| Oops. REMOVED NC.ProtectedSessionOperation wrapper)

(* * rht 3/30/87%: Now passes NIL InterestedWindow arg to NC.QuitCard so it will try to use card's prompt window.)

```
    (for Card in ActiveCards bind Window
       do [for CardPart in '(SUBSTANCE TOLINKS GLOBALTOLINKS PROPLIST)
             do (OR (NC.ApplyFn ObtainWritePermissionFn Card CardPart)
                    (until (NC.ApplyFn ObtainWritePermissionFn Card CardPart)
                       do (OR QuietFlg (NC.PrintMsg InterestedWindow T (CONCAT "Waiting to obtain write
                                                                               permission for the " CardPart " on
                                                                               card " (NC.FetchTitle Card)
                                                                               "...")))
                          (BLOCK)
                       finally (OR QuietFlg (NC.PrintMsg InterestedWindow NIL "done."]
          (NC.QuitCard Card T NIL T NIL NIL QuietFlg)
          (if (SETQ Window (NC.FetchWindow Card))
              then (bind (Process _ (WINDOWPROP Window 'PROCESS)) until (OR (NULL Process)
                                                                            (PROCESS.FINISHEDP Process))
                      do (BLOCK])
```

(NC.CleanupCardObjects

```
  [LAMBDA (HashArray)                                                (* fgh%: " 5-Sep-86 17:11")
```

(* * For every cardobject in HashArray, smash CardCache and UserData fields cause they might cause circular links.)

(* * fgh |9/5/86| Chack to make sure HashArray is real to avoid problems caused by MAPHASH of NIL.)

```
    (if (HARRAYP HashArray)
        then (MAPHASH HashArray (FUNCTION (LAMBDA (Card Key)
                                          (replace (Card CardCache) of Card with NIL)
                                          (replace (Card UserData) of Card with NIL])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.CloseNoteFileFns)
)

(RPAQ? NC.CloseNoteFileFns NIL)
```

;;; Checkpointing

```
(DEFINEQ
```

```
(NC.CheckpointDatabase
  [LAMBDA (NoteFile QuietFlg Don'tSaveDirtyCardsFlg InterestedWindow OperationMsg)
                                                                    (* Randy.Gobbel " 4-Mar-87 16:09")
```

(* * First save to the database any cards currently dirty. Copy the index array back into the file and set the LastChkptPtr to the end of the file.)

(* * rht 11/12/85%: Now calls NC.PutHashArray to do the hard work.)

(* * fgh |6/4/86| Added Don'tSaveDirtyCardsFlg to prevent double passes through active cards at close time)

(* * fgh |6/13/86| Changed printouts to NF menu. Added check for operations in progress.)

(* * fgh |6/25/86| Put in contention lock and NC.ProtectedNoteFileOperation Added Don'tCheckOperationInProgressFlg Don'tCheckCardOperationsInProgressFlg & InterestedWindow args.)

(* * rht 7/4/86%: Added check for readonly notefile.)

(* * rht 7/16/86%: Now passes QuietFlg arg down to NC.SaveDirtyCards.)

(* * fgh |9/1/86| Now just a compatibility wrapper for NC.CheckpointNoteFile.
Part of device vector impleementation.)

(* * rg |3/4/87| removed Don'tCheckXXXFlgs)

```
  (NC.CheckpointNoteFile NoteFile QuietFlg Don'tSaveDirtyCardsFlg InterestedWindow OperationMsg])

(NC.CheckpointNoteFile
  [LAMBDA (NoteFile QuietFlg Don'tSaveDirtyCardsFlg InterestedWindow OperationMsg)
                                                                    ; Edited  3-Dec-87 18:59 by rht:
```

(* * Checkpoint a notefile by call the device specific checkpoint fn.)

(* * fgh |5/26/86| First created.)

(* * fgh |9/1/86| Updated with with changes made to checkpinting since |5/23/86.|
Reimplemented changes include%: (* * fgh |6/4/86| Added Don'tSaveDirtyCardsFlg to prevent double passes through active cards at close time) (* * fgh |6/13/86| Changed printouts to NF menu.
Added check for operations in progress.) (* * fgh |6/25/86| Put in contention lock and NC.ProtectedNoteFileOperation Added Don'tCheckOperationInProgressFlg Don'tCheckCardOperationsInProgressFlg & InterestedWindow args.)
(* * rht 7/4/86%: Added check for readonly notefile.) (* * rht 7/16/86%: Now passes QuietFlg arg down to NC.SaveDirtyCards.))

(* * pmi |12/3/86| Added check for open NoteFile before attempting Checkpoint
(Code stolen from NC.CloseNoteFile))

(* * pmi |12/22/86| Made test for open notefile consistent with other NoteFile operations
(ie. Abort Close))

(* * rg |3/4/87| rewritten for new concurrency machinery)

(* * rht 3/25/87%: Now calls NC.CoerceToInterestedWindow.)

(* * rg |3/27/87| fiddled with NC.ProtectedNoteFileOperation wrapper)

(* * rht 4/3/87%: Took out extra "T" argument being passed to NC.SaveDirtyCards.)

```
  (DECLARE (GLOBALVARS NC.MsgDelay))
  (NC.ProtectedNoteFileOperation
   NoteFile "Checkpoint NoteFile" InterestedWindow (OR (OPENWP InterestedWindow)
                                                       (SETQ InterestedWindow (NC.CoerceToInterestedWindow
                                                                               NoteFile)))
   (PROG ((FullFileName (fetch (NoteFile FullFileName) of NoteFile))
          ReturnValue)
```

(* * Make sure NF is open)

```
         (if [NULL (ERSETQ (SETQ ReturnValue (NC.NoteFileOpenP NoteFile]
             then (SETQ ReturnValue 'OpenPFailed))
         (if (NULL ReturnValue)
             then
```

(* * NoteFile is not open.)

```
                 (NC.PrintMsg InterestedWindow T "Can't checkpoint a closed notefile." (CHARACTER 13))
                 (DISMISS NC.MsgDelay)
                 (NC.ClearMsg InterestedWindow T)
                 (RETURN NIL)
           elseif (NOT (type? NoteFile ReturnValue))
             then
```

(* * Error return from NoteFileOpenPFn)

```
                 (if [NULL (ERSETQ (NC.ReportError NIL (CONCAT "OpenP test on " FullFileName "failed because "
                                                               ReturnValue (CHARACTER 13)
```

```
                                                        "OK to continue Checkpoint. ^ to abort
                                                        Checkpoint."]
                    then (RETURN ReturnValue)))
          (SETQ OperationMsg (CONCAT (OR OperationMsg "")
                                     "Checkpointing " FullFileName (CHARACTER 13)))
          (if (NC.CheckForNotReadOnly NoteFile InterestedWindow "Can't checkpoint ")
              then (RETURN (PROGN [OR QuietFlg (RESETSAVE NIL `(NC.ClearMsg ,InterestedWindow T]

          (* * If appropriate, msg the user.)

                                    (OR QuietFlg (NC.PrintMsg InterestedWindow "Checkpointing notefile "
                                                              FullFileName " ..." (CHARACTER 13)))

          (* * Save the dirty cards on the screen if necessary.)

                                    (if (NULL Don'tSaveDirtyCardsFlg)
                                        then (NC.SaveDirtyCards NoteFile InterestedWindow OperationMsg QuietFlg))
                                                                    (* Put out the new ChkptPtr to the file.)

          (* * Call the device specific checkpoint fn.)

                                    (if [NULL (ERSETQ (SETQ ReturnValue (APPLY* (fetch (NoteFile
                                                                                       CheckpointNoteFileFn)
                                                                                   of NoteFile)
                                                                                NoteFile InterestedWindow
                                                                                OperationMsg QuietFlg]
                                        then (SETQ ReturnValue 'CheckpointFailed))

          (* * Process the error returns.)

                                    (if (type? NoteFile ReturnValue)
                                        then

          (* * Successful return.)

                                            (OR QuietFlg (NC.PrintMsg InterestedWindow T (OR OperationMsg "")
                                                                      " Checkpoint done."
                                                                      (CHARACTER 13)))
                                            NoteFile
                                      else

          (* * Error return.)

                                          (ERSETQ (NC.ReportError NIL (CONCAT "Checkpoint failed for "
                                                                              FullFileName " because " ReturnValue
                                                                              ".")))
                                          ReturnValue])
```

(NC.SaveDirtyCards

```
  [LAMBDA (NoteFile InterestedWindow OperationMsg QuietFlg)      (* Randy.Gobbel "30-Mar-87 18:10")
```

(* * Save every card that is both active and dirty to the notefile.)

(* * rht 9/21/85%: Now records cards that were shrunken and reshrinks after checkpoint is completed.)

(* * fgh |10/15/85| Put in stuff to make using cache array efficient)

(* * rht 10/20/85%: Now uses NC.GetShrunkenWin to find if card's win is shrunken, rather than looking at all open windows on the screen.)

(* * rht 11/13/85%: Updated to handle new NoteFile and Card formats.)

(* * fgh |6/13/86| Added TerminateOperationsInProgressFlg OperationMsg & InterestedWindow arg and associated action.)

(* * kirk/rht |9/10/86| Added QuietFlg)

(* * rht 3/30/87%: Now passes NIL InterestedWindow arg to NC.CardSaveFn so that card's prompt window will be used if possible.)

```
    (LET (ShrunkenCardWins ActiveCards)
         [NC.MapCards NoteFile (FUNCTION (LAMBDA (Card)
                                         (if (NC.ActiveCardP Card)
                                             then (push ActiveCards Card))
                                         (if (NC.GetShrunkenWin Card)
                                             then (push ShrunkenCardWins (NC.FetchWindow Card]
         (RESETLST
             (RESETSAVE NC.ForceSourcesFlg NIL)
             (RESETSAVE NC.ForceTitlesFlg NIL)
             (for Card in ActiveCards eachtime (BLOCK) do (NC.CardSaveFn Card QuietFlg NIL OperationMsg)))
         (for Win in ShrunkenCardWins eachtime (BLOCK) do (SHRINKW Win])

)
```

;;; Aborting an open NoteFile

```
(DEFINEQ
```

(NC.AbortSession

```
  [LAMBDA (NoteFile InterestedWindow Don'tConfirmFlg QuietFlg)    (* DSJ%: "29-Sep-87 00:59")
```

(* * Kill the current notecards session. Work lost since last checkpoint.)

(* * rht 7/14/85%: Replaced the call to reset the main menu with call to NC.ResetMainMenu.
Also took out redundant reset of PSA.Database, since NC.ForceDatabaseClose is doing that.)

(* * fgh & rht |10/16/85| Update with new cacheing mechanism.)

(* * fkr |11/8/85| Updated to handle noteFile object and new CardID scheme.)

(* * kirk 20Jan86 Added Don'tCloseFlg to leave NoteFile open after done deleting changes.)

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

(* * rht 7/2/86%: No longer bugs you if no changes were made since last checkpoint.
Removed Don'tCloseFlg arg and added InterestedWindow arg.)

(* * rht 7/6/86%: Now clears InterestedWindow of final truncating message.)

(* * rht 7/13/86%: Added Don'tConfirmFlg and QuietFlg args. Note that Don'tConfirmFlg non-nil stops questioning of user as to losing all changes.)

(* * kirk |11/17/86| Changed call on SETFILEINFO to pass stream instead of filename.)

(* * pmi |12/22/86| Made test for open notefile consistent with other NoteFile operations
(ie. Checkpoint, Close))

(* * rht 2/19/87%: Added DEL.PROCESS call to kill caching process.)

(* * rg |3/6/87| added NC.ProtectedSessionOperation wrapper)

(* * rht 3/25/87%: Now calls NC.CoerceToInterestedWindow.)

(* * rg |3/31/87| changed ProtectedSessionOp to ProtectedNoteFileOp)

(* * pmi 8/7/87%: Now asks for confirm no matter what work was done.)

(* * pmi 8/14/87%: Now calls NC.NoticeNoteFile to make sure the file has been noticed.)

(* * pmi 8/18/87%: No longer asks for confirm if notefile was open read-only.)

(* * dsj |9/23/87.| Added BEFORE AND AFTER call to NC.RunCloseEvents)

```
    (DECLARE (GLOBALVARS NC.MsgDelay))
    (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow NoteFile)))
    (NC.ProtectedNoteFileOperation
     NoteFile "Abort Session" InterestedWindow
     (PROG ((Stream (fetch (NoteFile Stream) of NoteFile))
            (FullFileName (fetch (NoteFile FullFileName) of NoteFile))
            (LastChkptPtr (fetch (NoteFile CheckptPtr) of NoteFile))
            EndPtr CardTotal NewBytes ReturnValue)
           (if [NULL (ERSETQ (SETQ ReturnValue (NC.NoteFileOpenP NoteFile]
               then (SETQ ReturnValue 'OpenPFailed))
           (if (NULL ReturnValue)
               then
```

(* * NoteFile is not open.)

```
                   (NC.PrintMsg InterestedWindow T "Can't abort a closed notefile." (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg InterestedWindow T)
                   (RETURN NIL)
             elseif (NOT (type? NoteFile ReturnValue))
               then
```

(* * Error return from NoteFileOpenPFn)

```
                   (if [NULL (ERSETQ (NC.ReportError NIL (CONCAT "OpenP test on " FullFileName "failed because "
                                                                 ReturnValue (CHARACTER 13)
                                                                 "OK to continue Abort. ^ to abort Abort."]
                       then (RETURN ReturnValue)))
```

(* * Delete the types and titles caching process if still alive. Have to do it now in order to make checking operations that follow suitably efficient. Note its a bit too early since we can still cancel this close.
But any harm done is loss of speed if NoteFile remains open when close iss cancelled.)

(* * dsj added this call to NC.RunCloseEvents Exit on a return of DON'T)

```
           (if (EQ (NC.RunCloseEvents NoteFile 'BEFORE)
                   'DON'T)
               then (RETURN NIL))
           (DEL.PROCESS (fetch (NoteFile CachingProcess) of NoteFile))
```

```
(* * Removed old confirm question based on amount of stuff written past the checkpoint.)

 (SETQ EndPtr (GETEOFPTR Stream))

(* * (SETQ NewBytes (IDIFFERENCE EndPtr LastChkptPtr)))

(* * This was in the if below%: (OR (ZEROP NewBytes) Don'tConfirmFlg
(NC.AskYesOrNo (CONCAT "Do you wish to lose all changes since"
(CHARACTER 13) "the last checkpoint (" NewBytes " bytes) of " FullFileName) "--" "Yes" T InterestedWindow NIL T)))

 (if (OR Don'tConfirmFlg (NC.ReadOnlyNoteFileP NoteFile)
         (NC.AskYesOrNo (CONCAT "Do you wish to lose all changes since" (CHARACTER 13)
                                "the open or last checkpoint of " FullFileName)
                "--" "No" T InterestedWindow NIL T))
     then [LET ((CardNumber 0)
                (CardTotal (fetch (NoteFile HashArraySize) of NoteFile)))
               (NC.MapCards NoteFile (FUNCTION (LAMBDA (Card)
                                                  (LET
                                                   (Win)
                                                   (SETQ CardNumber (ADD1 CardNumber))
                                                   [OR QuietFlg (COND
                                                                  ((ZEROP (IREMAINDER CardNumber 100))
                                                                   (NC.PrintMsg InterestedWindow T
                                                                          "Quitting from active cards
                                                                          ... " (CHARACTER 13)
                                                                          "Processing item number "
                                                                          CardNumber " out of "
                                                                          CardTotal "." (CHARACTER
                                                                                          13]
                                                   (COND
                                                      ((NC.ActiveCardP Card)
                                                       (SETQ Win (NC.FetchWindow Card))
                                                       (NC.AbortCard Card QuietFlg)
                                                       (COND
                                                          (Win (bind (Process _ (WINDOWPROP Win
                                                                                            'PROCESS))
                                                                  until (OR (NULL Process)
                                                                            (PROCESS.FINISHEDP Process))
                                                                  do (BLOCK))
                                                               (CLOSEW Win]
          [COND
             ((LESSP LastChkptPtr EndPtr)
              (OR QuietFlg (NC.PrintMsg InterestedWindow T "Truncating file " FullFileName " ..."))
              (COND
                 ((NOT (SETFILEINFO Stream 'LENGTH LastChkptPtr))
                  (NC.PrintMsg InterestedWindow NIL "Couldn't truncate " FullFileName "." (CHARACTER
                                                                                             13]
          (NC.ResetNoteFileInterface NoteFile)
          (NC.ForceDatabaseClose NoteFile)
          (NC.NoticeNoteFile NoteFile)

(* * added this call to NC.RunCloseEvents dsj)

          (if (EQ (NC.RunCloseEvents NoteFile 'AFTER)
                  'DON'T)
              then (RETURN NIL)))
 (NC.ClearMsg InterestedWindow T])
```

(NC.ForceDatabaseClose

```
 [LAMBDA (NoteFile Don'tMenuFlg)                                    (* pmi%: "19-Aug-87 15:06")
```

(* * Really close the database, i.e.. bypass the ADVISE on CLOSEF that prevents closing of the database.)

(* * rht 1/10/85%: Note new kludgey call to \UPDATEOF recommended by Tayloe to avoid truncation problems.)

(* * rht 2/5/85%: Added resetting of NC.UncachingNotCompleted here so it will happen after compact, repair, etc.)

(* * rht 7/9/85%: Added resetting of NC.LinkLabelsDate.)

(* * rht 11/10/85%: Updated to incorporate new NoteFile scheme.)

(* * kirk 31Dec85%: added Don'tMenuFlg)

(* * rht 1/8/86%: Now smashes old notefile object to remove cycles.
Don't you love interlisp gc'er?)

(* * rht 5/1/86%: Save Menu on notefile object when smashing.)

(* * rht 7/6/86%: Only closes notefile's stream if there is an open one.)

(* * fgh |9/1/86| Now saves the Device vector when cleaning up the NoteFile object.)

(* * pmi 5/19/87%: Replaced call to NC.RemoveNoteFileFromHashArray with NC.RemoveNoteFile as part of general cleanup.)

(* * pmi 8/19/87%: Added call calls to NC.NoticeNoteFile and NC.ResetNoteFileInterface.)

```
    (CLOSEF? (fetch (NoteFile Stream) of NoteFile))

          (* Smash the cardcache and userdata fields of all card objects for this notefile to remove circular links.)

    (ADD.PROCESS (LIST (FUNCTION NC.CleanupCardObjects)
                       (fetch (NoteFile HashArray) of NoteFile)))
    (replace (NoteFile Stream) of NoteFile with NIL)

          (* Smash the notefile object so we don't have cycles -
          card -> notefile -> card.)
                                                                    (* Usually we leave shell in notefiles hash array so there's a
                                                                    record.)
    (create NoteFile smashing NoteFile UID _ (fetch (NoteFile UID) of NoteFile)
                          FullFileName _ (fetch (NoteFile FullFileName) of NoteFile)
                          Menu _ (fetch (NoteFile Menu) of NoteFile)
                          NoteFileDevice _ (fetch (NoteFile NoteFileDevice) of NoteFile))

          (* * Reset the notefile menu icon to look closed.)

    (NC.ResetNoteFileInterface NoteFile)

          (* * Make sure the notefile has been noticed.)

    (NC.NoticeNoteFile NoteFile)
    (if Don'tMenuFlg
        then (NC.RemoveNoteFile NoteFile))
    NoteFile])

)


;;; Stuff to handle read-only notefiles.

(DEFINEQ

(NC.ReadOnlyNoteFileP
  [LAMBDA (NoteFile)                                                (* fgh%: " 1-Sep-86 13:40")

          (* * Return non-nil if notefile is open for read only.)

          (* * fgh |9/1/86| Reimplemented in accordance with device vector implementation.)

    (fetch (NoteFile ReadOnlyFlg) of NoteFile])

(NC.CheckForNotReadOnly
  [LAMBDA (CardOrNoteFile InterestedWindow OperationMsg)            (* rht%: "25-Mar-87 17:33")

          (* * If card or Notefile is read-only then print a message and return nil.
          Otherwise, return the card or notefile.)

          (* * rht 3/25/87%: Now calls NC.CoerceToInterestedWindow.)

    (OR (OPENWP InterestedWindow)
        (SETQ InterestedWindow (NC.CoerceToInterestedWindow CardOrNoteFile)))
    (if (COND
          ((NC.CardP CardOrNoteFile)
           (NC.ReadOnlyCardP CardOrNoteFile))
          ((type? NoteFile CardOrNoteFile)
           (NC.ReadOnlyNoteFileP CardOrNoteFile)))
        then (NC.PrintMsg InterestedWindow T OperationMsg "a notefile open for read-only." (CHARACTER 13))
             (DISMISS 1500)
             (NC.ClearMsg InterestedWindow T)
             NIL
      else CardOrNoteFile])

)


;;; Other database operations.
;; see also NCCOMPACT

(DEFINEQ

(NC.DeleteDatabaseFile
  [LAMBDA (FileNameOrNoteFile InterestedWindow Don'tConfirmFlg QuietFlg)
                                                                    (* pmi%: "29-May-87 18:24")
                                                                    (* Delete file FileName)

          (* * rht 8/7/84%: If delete happens, clear NC.DatabaseFileNameSuggestion.)

          (* * rht 3/17/85%: Fixed for case when user specifies version number of file to delete.)

          (* * fkr 11/8/85%: Ripped out PSA.Database check. Added check for file open.)

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

```
        (* * fgh |6/24/86| Added ability to pass down NoteFile object as well as file name.
        Added code to remove NF from NFs hash array and remove the menu on the screen.)

        (* * rht 7/2/86%: No longer prints completed message with DISMISS.
        Now returns non-nil if successful. Accepts Don'tConfirmFlg arg.)

        (* * fgh |7/5/86| Added call to RemoveAccessToNoteFile.)

        (* * rht 7/13/86%: Added QuietFlg arg.)

        (* * kef 8/8/86%: Factored out into device specific vectors.)

        (* * pmi |12/3/86| Added check for open file)

        (* * pmi |12/19/86| Changed to call NC.NoteFileOpenP instead of OPENP to check whether the file is open.
        Added NC.DeviceVectorsHashArray. to GLOBALVARS)

        (* * pmi 5/29/87%: Fixed search for full filename to work for non-DSK files.
        Added error message if delete fails. Added test for existence of file to be deleted.)

    (DECLARE (GLOBALVARS NC.MsgDelay NC.DeviceVectorsHashArray))
    (PROG ((FileName (if (type? NoteFile FileNameOrNoteFile)
                         then (fetch (NoteFile FullFileName) of FileNameOrNoteFile)
                       else FileNameOrNoteFile))
           (MsgWindow (OR InterestedWindow (NC.CoerceToInterestedWindow FileNameOrNoteFile)))
           FullFileName)                                          (* Make sure no open databases)
                                                                  (* Get file name)
          (AND (NULL FileName)
               (NULL (SETQ FileName (NC.DatabaseFileName "Name of Notefile to be deleted:" " -- " T NIL NIL
                                          MsgWindow)))
               (RETURN NIL))

          (* make sure to-be-deleted file exists and that we get the earliest version, if version is not specified.)

          [SETQ FullFileName (if (FILENAMEFIELD FileName 'VERSION)
                                 then (FULLNAME FileName)
                               else (CAR (FILDIR-EARLIEST FileName]

          (* * Can't delete a non-existent file.)

          (if (NULL FullFileName)
              then (NC.RemoveAccessToNoteFile FileName)
                   (SETQ MsgWindow (NC.CoerceToInterestedWindow MsgWindow))
                   (NC.PrintMsg (NC.AttachPromptWindow MsgWindow)
                          T FileName " does not exist." (CHARACTER 13)
                          "Delete cancelled."
                          (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg MsgWindow T)
                   (RETURN))

          (* * Don't try to delete if the NoteFile is open)

          (if (NC.NoteFileOpenP FullFileName)
              then (NC.PrintMsg MsgWindow T "Can't delete an open notefile." (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg MsgWindow T)
                   (RETURN NIL))

          (* * Ask user to confirm twice.)

          (OR Don'tConfirmFlg (if (NOT (NC.AskYesOrNo (CONCAT "Are you sure you want to delete " (CHARACTER 13)
                                                               FullFileName "?" (CHARACTER 13))
                                                       " -- " "No" T (NC.AttachPromptWindow MsgWindow)
                                                       (NOT MsgWindow)))
                                  then (if QuietFlg
                                           else (NC.PrintMsg (NC.AttachPromptWindow MsgWindow)
                                                             T FullFileName " not deleted." (CHARACTER 13))
                                                (DISMISS NC.MsgDelay)
                                                (NC.ClearMsg MsgWindow T))
                                       (RETURN)))
          (OR QuietFlg (NC.PrintMsg (NC.AttachPromptWindow MsgWindow)
                                    T "Deleting" FullFileName (CHARACTER 13)))
          (DISMISS 1000)
          (OR Don'tConfirmFlg (if (NOT (NC.AskYesOrNo (CONCAT "Are you still sure you want to delete "
                                                               (CHARACTER 13)
                                                               FullFileName "?" (CHARACTER 13))
                                                       " -- " "No" T (NC.AttachPromptWindow MsgWindow)
                                                       (NOT MsgWindow)))
                                  then (OR QuietFlg (NC.PrintMsg (NC.AttachPromptWindow MsgWindow)
                                                                 T FullFileName " not deleted." (CHARACTER 13)))
                                       (RETURN)))

          (* * Delete the file)

          (SETQ FullFileName (APPLY* (fetch (NoteFileDevice DeleteNoteFileFn) of (GETHASH (COND
```

```
                                                                    ((NC.RemoteHostP
                                                                      FullFileName)
                                                                     'REMOTEMULTIUSER)
                                                                    (T 'LOCALSINGLEUSER)
                                                                    )
                                                            NC.DeviceVectorsHashArray
                                                            ))
                                    FullFileName))
        (if (NULL FullFileName)
            then (NC.PrintMsg (NC.AttachPromptWindow MsgWindow)
                      T FileName " could not be deleted." (CHARACTER 13)
                      "Delete cancelled."
                      (CHARACTER 13))
                 (DISMISS NC.MsgDelay)
                 (NC.ClearMsg MsgWindow T)
                 (RETURN))
        (NC.RemoveAccessToNoteFile FullFileName)
        (SETQ NC.DatabaseFileNameSuggestion NIL)
        (NC.ClearMsg MsgWindow T)
        (RETURN FullFileName])
```

(NC.CopyNoteFile

```
  [LAMBDA (FromNoteFileOrName ToFileName InterestedWindow)       (* pmi%: "29-May-87 19:01")
```

(* * Copy a notefile. Ask user for names of FromNoteFileOrName and ToFileName.)

(* * fkr 11/8/85%: Ripped out PSA.Database check. Now takes FromNoteFileOrName and ToFileName args.)

(* * kirk 19May86 Fixed to work from NoteFile menu)

(* * fgh |6/24/86| Fixed bug where would not work if NULL FromNoteFileOrName)

(* * fgh |7/5/86| Closes prompt window when done.)

(* * rht 11/1/86%: Added missing var binding.)

(* * pmi |12/19/86| Changed to call NC.NoteFileOpenP instead of OPENP to check whether the file is open.)

(* * pmi 5/29/87%: Added call to NC.NoticeNoteFile to make sure From NoteFile is noticed.
Added call to NC.NoticeNoteFileName for the To file name. Cleaned up case where notefilename is valid, but a notefile object does not exist.)

```
    (DECLARE (GLOBALVARS NC.MsgDelay NC.DatabaseFileNameSuggestion))
    (PROG ((MsgWindow InterestedWindow)
           Result FromFileName FullFromFileName FromNoteFile)
```

(* * Get the name of the file to be copied.)

```
          (if (type? NoteFile FromNoteFileOrName)
              then (SETQ FromNoteFile FromNoteFileOrName)
                   (SETQ FromFileName (fetch (NoteFile FullFileName) of FromNoteFile))
            elseif (SETQ FromFileName (OR FromNoteFileOrName (NC.DatabaseFileName "Name of NoteFile to be copied:"
                                                                                  " -- " T NIL NIL MsgWindow)))
            else (RETURN NIL))
```

(* * Check for existence of file to be copied.)

```
          (if (SETQ FullFromFileName (FULLNAME FromFileName))
              then [if (OR FromNoteFile (SETQ FromNoteFile (NC.NoteFileFromFileName FullFromFileName)))
                       then (SETQ MsgWindow (OR MsgWindow (NC.CoerceToInterestedWindow FromNoteFile]
            else (NC.RemoveAccessToNoteFile FromFileName)
                 (SETQ MsgWindow (NC.CoerceToInterestedWindow MsgWindow))
                 (NC.PrintMsg MsgWindow T FromFileName " does not exist." (CHARACTER 13)
                        "Copy cancelled."
                        (CHARACTER 13))
                 (DISMISS NC.MsgDelay)
                 (NC.ClearMsg MsgWindow T)
                 (RETURN NIL))
```

(* * Check to see if the notefile is open, abort if it is.)

```
          (if (NC.NoteFileOpenP FromNoteFile)
              then (NC.PrintMsg MsgWindow T "Can't copy an open notefile." (CHARACTER 13)
                        "Copy cancelled."
                        (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg MsgWindow T)
                   (RETURN NIL))
```

(* * Add the From notefile to the Hash Array and the list of noticed notefiles, in case it isn't already there.)

```
          (NC.NoticeNoteFile FullFromFileName)
          (OR ToFileName (SETQ ToFileName (NC.DatabaseFileName "Name of target of copy:" " -- " T NIL NIL
                                               MsgWindow))
              (RETURN))
          (if (AND FullFromFileName ToFileName)
```

```
            then (NC.PrintMsg MsgWindow T "Copying " FullFromFileName " to " ToFileName " ...")
                 (COND
                    ((SETQ Result (COPYFILE FullFromFileName ToFileName))
                     (NC.PrintMsg MsgWindow T FullFromFileName " copied to " Result "." (CHARACTER 13))

          (* * Notice the To NoteFile name)

                     (NC.NoticeNoteFileName Result)
                     (SETQ NC.DatabaseFileNameSuggestion (PACKFILENAME 'VERSION NIL 'BODY Result))
                     (DISMISS NC.MsgDelay)
                     (NC.ClearMsg MsgWindow T)))
            elseif (NULL FullFromFileName)
              then (NC.PrintMsg MsgWindow T "Can't open file for copy: " FromNoteFileOrName (CHARACTER 13)
                           "Copy cancelled."
                           (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg MsgWindow T])
```

(NC.RenameNoteFile

```
[LAMBDA (FromNoteFileOrName ToFileName InterestedWindow)       (* pmi%: "29-May-87 18:46")
```

(* * Rename a notefile. Ask user for names of FromNoteFileOrName and ToFileName.)

(* * fgh |7/5/86| First created on basis of CopyNoteFile.)

(* * rht 11/1/86%: Added missing var binding.)

(* * pmi |12/19/86| Changed to call NC.NoteFileOpenP instead of OPENP to check whether the file is open.
Made consistent with other NoteFile operations in the way it checks for valid NoteFile, gets msg window, etc.)

(* * pmi 5/29/87%: Added calls to notice the new file name and to forget the old file name.
Cleaned up case where notefilename is valid, but a notefile object does not exist.)

```
  (DECLARE (GLOBALVARS NC.MsgDelay NC.DatabaseFileNameSuggestion))
  (PROG ((MsgWindow InterestedWindow)
         Result NoteFile FromFileName FullFromFileName)

          (* * Get the name of the file to be compacted)

          (if (type? NoteFile FromNoteFileOrName)
              then (SETQ NoteFile FromNoteFileOrName)
                   (SETQ FromFileName (fetch (NoteFile FullFileName) of NoteFile))
            elseif (SETQ FromFileName (OR FromNoteFileOrName (NC.DatabaseFileName "Name of NoteFile to be
                                                                                   renamed:" " -- " T NIL NIL MsgWindow)))
            else (RETURN NIL))

          (* * Check for existence of file to be renamed.)

          (if (SETQ FullFromFileName (FULLNAME FromFileName))
              then [if (OR NoteFile (SETQ NoteFile (NC.NoteFileFromFileName FullFromFileName)))
                       then (SETQ MsgWindow (OR MsgWindow (NC.CoerceToInterestedWindow NoteFile]
            else (NC.RemoveAccessToNoteFile FromFileName)
                 (SETQ MsgWindow (NC.CoerceToInterestedWindow MsgWindow))
                 (NC.PrintMsg MsgWindow T FromFileName " does not exist." (CHARACTER 13)
                        "Rename cancelled."
                        (CHARACTER 13))
                 (DISMISS NC.MsgDelay)
                 (NC.ClearMsg MsgWindow T)
                 (RETURN NIL))

          (* * Check to see if the notefile is open, abort if it is.)

          (if (NC.NoteFileOpenP NoteFile)
              then (NC.PrintMsg MsgWindow T "Can't rename an open notefile." (CHARACTER 13)
                           "Rename cancelled."
                           (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg MsgWindow T)
                   (RETURN NIL))
          (OR ToFileName (SETQ ToFileName (NC.DatabaseFileName "New name for the NoteFile:" " -- " T NIL NIL
                                               MsgWindow))
              (RETURN))
          (if (AND FullFromFileName ToFileName)
              then (NC.PrintMsg MsgWindow T "Renaming " FullFromFileName " to " ToFileName " ...")
                   (if (SETQ Result (RENAMEFILE FullFromFileName ToFileName))
                       then (NC.PrintMsg MsgWindow T FullFromFileName " renamed to " Result "." (CHARACTER 13))

          (* * Fix up the NoteFile with the new name.)

                            (AND NoteFile (replace (NoteFile FullFileName) of NoteFile with Result)
                                 (AND (fetch (NoteFile Menu) of NoteFile)
                                      (WFROMMENU (fetch (NoteFile Menu) of NoteFile))
                                      (NC.SetUpNoteFileInterface NoteFile)))

          (* * Forget the old file name)
```

```
                        (NC.RemoveNoteFileName FullFromFileName)

          (* * Register the new file name as noticed.)

                        (NC.NoticeNoteFileName Result)
                        (SETQ NC.DatabaseFileNameSuggestion (PACKFILENAME 'VERSION NIL 'BODY Result))
                        (DISMISS NC.MsgDelay)
                        (NC.ClearMsg MsgWindow T))
            elseif (NULL FullFromFileName)
              then (NC.PrintMsg MsgWindow T "Can't open file for rename: " FromNoteFileOrName (CHARACTER 13)
                           "Rename cancelled."
                           (CHARACTER 13))
                   (DISMISS NC.MsgDelay)
                   (NC.ClearMsg MsgWindow T])
```

(NC.RemoveAccessToNoteFile

```
  [LAMBDA (NoteFileOrFileNameOrUID)                                    (* pmi%: "21-May-87 11:49")

          (* * Remove a NoteFile from the NoteFiles hash array and close its menu if its on the screen.)

          (* * pmi 5/21/87%: Cleaned up to call NC.RemoveNoteFile for the hash array stuff.)

    (LET ((NoteFileObject (NC.RemoveNoteFile NoteFileOrFileNameOrUID))
          NoteFileName Menu)
         [SETQ Menu (if NoteFileObject
                        then (SETQ NoteFileName (fetch (NoteFile FullFileName)
                                                   NoteFileObject))
                             (OR (fetch (NoteFile Menu) of NoteFileObject)
                                 (GETPROP NoteFileName 'Menu))
                        else (GETPROP (SETQ NoteFileName NoteFileOrFileNameOrUID)
                                      'Menu]
         (if Menu
             then (CLOSEW (WFROMMENU Menu))
                  (SETQ Menu])

)


;;; Functions for getting stuff off the notefile. These manipulate file pointer so run with monitor lock.

(DEFINEQ
```

(NC.GetNoteCard

```
  [LAMBDA (Card)                                                       (* rht%: "16-Oct-86 18:07")

          (* * Get a note card from the database.)

          (* * kirk 27Nov85 Changed to call NC.GetMainCardData)

          (* * rht 10/16/86%: Now checks that card is active before doing Get.)

    (if (EQ (fetch (Card Status) of Card)
            'ACTIVE)
        then (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
                  (PROGN (NC.GetMainCardData Card)
                         (NC.GetLinks Card)
                         (NC.GetTitle Card)
                         (NC.GetPropList Card)
                         (NC.ActivateCard Card)
                         Card))])
```

(NC.GetMainCardData

```
  [LAMBDA (Card OverrideStream)                                        ; Edited 14-Dec-88 15:29 by krivacic
    ;; Get a note card from the database.  If IncludeDeletedCardsFlg is NIL, then return immediately if card is deleted or free.  Otherwise, get dekleted
    ;; but not free cards.

;;; rht 1/31/85: Now reads pointers from index array rather than file.

;;; rht 7/9/85: Now gets date if notefile has newer data format.

;;; rht 11/10/85 Updated to handle new Card scheme and NoteFile objects.

;;; fgh 11/20/85 Added call to NC.ReadCardPartHeader and put in code to read Start and End pointers before calling card type's getfn.

;;; kirk 27Nov85 abstracted this function out of NC.GetNoteCard

;;; rht 1/23/86: Now takes optional OverrideStream arg.  This, if given, overrides stream of card's notefile.

;;; rht 1/28/86: Now passes extra arg to NC.ReadCardPartHeader indicating that when we're overriding the notefile stream, you shouldn't force UIDs on
;;; stream and in card to match.

;;; fgh 2/5/86 Added call to NC.ApplyFn

;;; fgh 2/6/86 Added support for version numbers on the substance get fn.
```

;;; kirk 14Feb86 Merged the above 4 changes

;;; kef 7/16/86: Uses the device vector GetCardPartFn to set up the stream and stream pointer for reading.

;;; kef 8/1/86: Moved the check for ACTIVE status to beginning.

;;; fgh 8/31/86 Adpated to use NC.DoCardPartFn.

;;; pmi 11/4/86 Reinstated Randy's changes (1/23/86 and 1/28/86) which somehow got lost.

;;; rht 3/13/87: No longer goes to notefile if card hasn't been saved yet.

```
    (DECLARE (GLOBALVARS NC.ItemIdentifier))
    (if (AND (EQ (fetch (Card Status) of Card)
                 'ACTIVE)
             (NOT (NC.FetchNewCardFlg Card)))
        then (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
                 (NC.DoCardPartFn Get Card 'SUBSTANCE (LET ((Stream (OR (STREAMP OverrideStream)
                                                                        (NC.CoerceToNoteFileStream Card)))
                                                            Length SubstanceVersion)

;;; Read the header info

                                                           (NC.SetItemDate Card (NC.ReadCardPartHeader Card
                                                                                       NC.ItemIdentifier Stream
                                                                                       OverrideStream))

;;; read card type and region

                                                           (NC.SetType Card (NC.ReadCardType Stream))
                                                           (NC.SetRegion Card (NC.ReadRegion Stream))

;;; Read the length of substance, then call the substance get fn

                                                           (SETQ Length (NC.ReadPtr Stream 3))
                                                           (SETQ SubstanceVersion (NC.GetPtr Stream 1))
                                                           (NC.SetSubstance Card (NC.ApplyFn GetFn Card Length
                                                                                             Stream SubstanceVersion)
                                                                  )
                                                           Card)))])
```

(NC.GetLinks

```
  [LAMBDA (Card OverrideStream)                                    (* rht%: "13-Mar-87 17:06")
```

(* * rht 1/31/85%: Now reads pointers from index array.)

(* * rht 2/9/85%: Now fixes display formats on links read in.)

(* * rht 7/9/85%: Now gets date if notefile has newer data format.)

(* * fkr |11/8/85| Updated to handle new Card scheme and NoteFile objects.)

(* * fgh |11/20/85| Added call to NC.ReadCardPartHeader)

(* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

(* * rht 1/28/86%: Now passes extra arg to NC.ReadCardPartHeader indicating that when we're overriding the notefile stream, you shouldn't force UIDs on stream and in card to match.)

(* * kef 7/16/86%: Uses the device vector GetCardPartFn to set up the stream and stream pointer for reading.)

(* * kef 8/1/86%: Moved the check for ACTIVE status to beginning.)

(* * fgh |8/31/86| Adapted to use NC.DoCardPartFn.)

(* * pmi |11/4/86| Reinstated Randy's changes (|1/23/86| and |1/28/86|) which somehow got lost.)

(* * rht 3/13/87%: No longer goes to notefile if card hasn't been saved yet.)

```
    (DECLARE (GLOBALVARS NC.LinksIdentifier))
    (if (AND (EQ (fetch (Card Status) of Card)
                 'ACTIVE)
             (NOT (NC.FetchNewCardFlg Card)))
        then (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
                 (NC.DoCardPartFn Get Card 'LINKS (LET [(Stream (OR (STREAMP OverrideStream)
                                                                    (NC.CoerceToNoteFileStream Card]
```

(* * Read the header and set the date)

```
                                                       (NC.SetLinksDate Card (NC.ReadCardPartHeader Card
                                                                                   NC.LinksIdentifier Stream
                                                                                   OverrideStream))
```

(* * Read the links)

```
                                                       (NC.SetToLinks Card (NC.ReadListOfLinks Stream))
```

```
                                                    (NC.SetFromLinks Card (NC.ReadListOfLinks Stream))
                                                    (NC.SetGlobalLinks Card (NC.ReadListOfLinks Stream))
                                                    (NC.SetLinksDirtyFlg Card NIL)
                                                    Card))))])
```

(NC.GetTitle

```
[LAMBDA (Card OverrideStream)                                   (* rht%: "13-Mar-87 17:06")
                                                                (* Retrieve title for card specified by Card from the database
                                                                specified by DatabaseStream)
```

(* * rht 1/31/85%: Now reads pointers from index array rather than file.)

(* * rht 7/9/85%: Now gets date if notefile has newer data format.)

(* * kirk |10/28/85| Now returns NIL if Status not ACTIVE)

(* * fkr 10/29/85%: Fixed to use new numeric ID format.)

(* * rht |11/10/85| Updated to handle new Card scheme and NoteFile objects.)

(* * fgh |11/20/85| Added call to NC.ReadCardPartHeader)

(* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

(* * rht 1/28/86%: Now passes extra arg to NC.ReadCardPartHeader indicating that when we're overriding the notefile stream, you shouldn't force UIDs on stream and in card to match.)

(* * kef 7/16/86%: Uses the device vector GetCardPartFn to set the stream and stream pointer in preparation for the read.)

(* * kef 7/24/86%: Added check of NewCardFlg.)

(* * fgh |8/31/86| Adtaped to use NC.DoCardPartFn.)

(* * pmi |11/4/86| Reinstated Randy's changes (|1/23/86| and |1/28/86|) which somehow got lost.)

(* * rht 3/13/87%: No longer goes to notefile if card hasn't been saved yet.)

```
(DECLARE (GLOBALVARS NC.TitlesIdentifier))
(if (AND (EQ (fetch (Card Status) of Card)
             'ACTIVE)
         (NOT (fetch (Card NewCardFlg) of Card)))
    then (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
                       (NC.DoCardPartFn Get Card 'TITLE
```

(* * Now we've actually executed the BEFORE part of the GETFN, while in a RESETLST that will ensure execution of the AFTER part upon exit. This means that right now the Stream slot of the NoteFile is a random access stream with the file pointer set to the beginning of the card part.)

```
                    (LET ((Stream (OR (STREAMP OverrideStream)
                                      (NC.CoerceToNoteFileStream Card)))
                          Title)
                         (NC.SetTitleDate Card (NC.ReadCardPartHeader Card NC.TitlesIdentifier Stream
                                                                      OverrideStream))
                         (NC.SetTitle Card (SETQ Title (NC.ReadTitle Stream)))
                         Title))])
```

(NC.GetPropList

```
[LAMBDA (Card OverrideStream)                                   (* rht%: "13-Mar-87 17:03")
```

(* Retrieve the prop list for card specified by ID from the database specified by DatabaseStream)

(* * rht 1/31/85%: Now reads pointers from index array rather than file.)

(* * rht 7/9/85%: Now gets date if notefile has newer data format.)

(* * rht |11/10/85| Updated to handle new Card scheme and NoteFile objects.)

(* * fgh |11/20/85| Added call to NC.ReadCardPartHeader)

(* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

(* * rht 1/28/86%: Now passes extra arg to NC.ReadCardPartHeader indicating that when we're overriding the notefile stream, you shouldn't force UIDs on stream and in card to match.)

(* * kef 7/16/86%: Uses the device vector GetCardPartFn to set up the stream and stream pointer for reading.)

(* * kef 8/1/86%: Moved the check for ACTIVE status to beginning.)

(* * fgh |8/31/86| Adapted to use NC.DoCardPartFn.)

(* * pmi |11/4/86| Reinstated Randy's changes (|1/23/86| and |1/28/86|) which somehow got lost.)

(* * rht 3/13/87%: No longer goes to notefile if card hasn't been saved yet.)

```
(DECLARE (GLOBALVARS NC.PropsIdentifier))
```

```
    (if (AND (EQ (fetch (Card Status) of Card)
                 'ACTIVE)
             (NOT (NC.FetchNewCardFlg Card)))
        then (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
                   (NC.DoCardPartFn Get Card 'PROPLIST (LET ((Stream (OR (STREAMP OverrideStream)
                                                                         (NC.CoerceToNoteFileStream Card)))
                                                             Props)

          (* * set the fileptr to the beginning of the data, read the header, then read the prop list)

                                                             (NC.SetPropListDate Card (NC.ReadCardPartHeader Card
                                                                                             NC.PropsIdentifier
                                                                                             Stream OverrideStream)
                                                                   )
                                                             (NC.SetPropList Card (SETQ Props (NC.ReadPropList Stream
                                                                                                               )))
                                                             Props)))])
```

(NC.GetType

```
  [LAMBDA (Card)                                                    (* rht%: "30-May-87 15:55")
```
(* Retrieve the NoteCardType of card specified by NoteCardID from the database specified by DatabaseStream)

```
          (* * rht 1/31/85%: Now reads pointers from index array rather than file.)

          (* * rht 7/9/85%: Now gets date if notefile has newer data format.)

          (* * kirk 10/18/85; Now returns NIL if status not ACTIVE)

          (* * fkr 10/29/85%: Fixed to use new numeric ID format. Also added NC.SetTitle call.)

          (* * rht |11/10/85| Updated to handle new Card scheme and NoteFile objects.)

          (* * fgh |11/20/85| Added call to NC.ReadCardPartHeader)

          (* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

          (* * rht 1/28/86%: Now passes extra arg to NC.ReadCardPartHeader indicating that when we're overriding the notefile
          stream, you shouldn't force UIDs on stream and in card to match.)

          (* * kef 7/16/86%: Uses the device vector GetCardPartFn to set up the stream and stream pointer for reading.)

          (* * kef 7/30/86%: Changed to use GetCardInfoFn of the device vector.)

          (* * kef 8/1/86%: Decided there are some cases, such as new cards, where we may want the Type of the card even though
          it isn't ACTIVE yet.)

          (* * fgh |8/31/86| Changed APPLY* to NC.ApplyFn.)

          (* * rht 5/30/87%: No longer goes to notefile unless status is ACTIVE and card isn't brand new.)

    (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
        (if (AND (EQ (NC.FetchStatus Card)
                     'ACTIVE)
                 (NOT (fetch (Card NewCardFlg) of Card)))
            then (LET (CardType)
                      [NC.SetType Card (SETQ CardType (CDR (FASSOC 'TYPE (NC.ApplyFn GetCardInfoFn Card
                                                                                    'TYPE]
                      CardType)))])
```

(NC.GetSpecialCards

```
  [LAMBDA (NoteFile QuiteFlg InterestedWindow OperationMsg)      (* fgh%: "27-Jun-86 20:29")
```

          (* * fgh |1/16/86| Activate but don't display the special cards. This essentially caches them for fast access.)

          (* * fgh |6/27/86| Added InterestedWindow & OperationMsg args.)

```
    (OR QuiteFlg (NC.PrintMsg InterestedWindow T OperationMsg "Loading top level cards ..." (CHARACTER 13)))
    (NCP.ActivateCards (LIST (fetch (NoteFile TableOfContentsCard) of NoteFile)
                             (fetch (NoteFile OrphansCard) of NoteFile)
                             (fetch (NoteFile ToBeFiledCard) of NoteFile)
                             (fetch (NoteFile LinkLabelsCard) of NoteFile)
                             (fetch (NoteFile RegistryCard) of NoteFile])

)

(DEFINEQ
```

(NC.FetchSpecialCards

```
  [LAMBDA (NoteFile)                                                (* rht%: " 2-Mar-86 15:07")
```

          (* * Return list of special cards.)

```
    (LIST (fetch (NoteFile TableOfContentsCard) of NoteFile)
          (fetch (NoteFile OrphansCard) of NoteFile)
```

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

)

;;; Functions for putting stuff on the notefile. These manipulate file pointer so run with monitor lock.

(DEFINEQ

(NC.PutNoteCard
  [LAMBDA (Card UpdateUpdateListFlg UseOldDateFlg)                    (* rht%: "17-Feb-86 16:11")

          (* * Put down each of the card's parts to its notefile.)

    (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
        (NC.PutMainCardData Card UpdateUpdateListFlg UseOldDateFlg)
        (NC.PutLinks Card UseOldDateFlg)
        (NC.PutTitle Card UseOldDateFlg)
        (NC.PutPropList Card UseOldDateFlg))])

(NC.PutMainCardData
  [LAMBDA (Card UpdateUpdateListFlg UseOldDateFlg OverrideStream)
                                                                      (* Randy.Gobbel "10-Jun-87 17:34")

          (* * Write note card specified by ID to the database specified by Database stream)

          (* * rht 7/9/85%: Now puts out date after identifier. If UseOldDateFlg is non-nil, then use old date, otherwise use current
          date.)

          (* * rht 11/10/85%: Updated to handle NoteFile and Card scheme.)

          (* * fgh |11/20/85| Added call to NC.WriteCardPartHeader and the mechanism to write the start and end pointers of the
          substance before calling the card type's putfn.)

          (* * kirk 29Nov85 Renamed from NC.PutNoteCard)

          (* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

          (* * fgh |2/5/86| Added call to NC.ApplyFn)

          (* * fgh |2/6/86| Added support for version numbers on the substance put fn.)

          (* * kirk 14Feb86 Merged above two changes)

          (* * rht 2/14/86%: Fixed so call to NC.WriteCardType takes Stream as arg.)

          (* * rht 2/17/86%: Fixed so calls to NC.WriteCardPartHeader and to NC.WriteRegion take Stream arg.)

          (* * kef 7/16/86%: Makes use of the NoteFile device vector PutCardPartFn.)

          (* * kef 8/1/86%: Added notification of status change.)

          (* * fgh |8/31/86| Adapted to use NC.DoCardPartFn.)

          (* * pmi |11/4/86| Reinstated Randy's change (|1/23/86|) which somehow got lost.)

          (* * rht 11/14/86%: Now makes sure hung var PutSuccessfulLoc is NIL if we were passed an OverrideStream.)

          (* * rg |6/10/87| adds links cache if none already exists)

    (DECLARE (GLOBALVARS NC.ItemIdentifier))
    (LET (PutSuccessfulLoc)
         (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
             (NC.DoCardPartFn Put Card 'SUBSTANCE (LET ((Stream (OR (STREAMP OverrideStream)
                                                                    (NC.CoerceToNoteFileStream Card)))
                                                        StartDataLoc EndLoc CardType StartSubstanceLoc
                                                        SubstanceVersion)

          (* * Record update date on update list if necessary.)

                                                       (AND UpdateUpdateListFlg (NC.UpdateUpdateList Card))

          (* * First write out the card part header)

                                                       (SETQ StartDataLoc (GETFILEPTR Stream))
                                                       (NC.WriteCardPartHeader Card NC.ItemIdentifier
                                                              [COND
                                                                 (UseOldDateFlg (NC.FetchItemDate Card))
                                                                 (T (NC.SetItemDate Card (DATE]
                                                              Stream)

          (* * write out the type and region)

                                                       (NC.WriteCardType Stream (SETQ CardType (NC.RetrieveType
```

```
                                                                    Card)))
                                                (NC.WriteRegion Card Stream)
```

(* * Write out the dummy length pointer for and version byte the actual substance)

```
                                                (SETQ StartSubstanceLoc (GETFILEPTR Stream))
                                                (NC.WritePtr Stream 0 4)
```

(* * Write out the substance of the card.)

```
                                                (SETQ SubstanceVersion (NC.ApplyFn PutFn Card Stream))
```

(* * Update the length pointer at beginning of substance Subtract four so that length is the length of the actual substance and doesn't include the length pointer and version byte maintained here.
Also updated the version number returned by the put fn.)

```
                                                (SETQ EndLoc (GETFILEPTR Stream))
                                                (SETFILEPTR Stream StartSubstanceLoc)
                                                (NC.WritePtr Stream (DIFFERENCE (DIFFERENCE EndLoc
                                                                                            StartSubstanceLoc)
                                                                                4)
                                                       3)
                                                (SETQ SubstanceVersion (OR SubstanceVersion 0))
                                                (NC.WritePtr Stream SubstanceVersion 1)
```

(* * Update the length field at the beginning of the card info)

```
                                                (SETFILEPTR Stream StartDataLoc)
                                                (NC.WritePtr Stream (DIFFERENCE EndLoc StartDataLoc)
                                                       3)
                                                (SETFILEPTR Stream EndLoc)
```

(* * Now update the Index to reflect the new data just written. Done last in case the substance putting bombed for some reason.)

```
                                                (replace (Card Status) of Card with 'ACTIVE)
                                                         (* cause links cache to be created if not already in existence)
                                                (OR (fetch (Card Links) of Card)
                                                    (replace (Card FromLinks) of Card with NIL))
```

(* * Don't put a reasonable value in the hung variable PutSuccessfulLoc if we were passed OverrideStream.)

```
                                                (SETQ PutSuccessfulLoc (if OverrideStream
                                                                           then NIL
                                                                         else StartDataLoc))
                                               Card)))])
```

(NC.PutLinks

```
  [LAMBDA (Card UseOldDateFlg OverrideStream)                    (* rht%: " 1-Jul-87 10:23")
```

(* * Put the link data for ID onto the database file.)

(* * rht 1/30/85%: Changed to use index array instead of file.)

(* * rht 7/9/85%: Now puts out date after identifier. If UseOldDateFlg is non-nil, then use old date, otherwise use current date.)

(* * rht 11/10/85%: Updated to handle NoteFile and CardID scheme.)

(* * fgh |11/20/85| Added call to NC.WriteCardPartHeader and the mechanism to write the start and end pointers of the substance before calling the card type's putfn.)

(* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

(* * kef 7/16/86%: Makes use of the NoteFile device vector PutCardPartFn.)

(* * fgh |8/31/86| Adapated to use NC.DoCardPartFn.)

(* * pmi |11/4/86| Reinstated Randy's change (|1/23/86|) which somehow got lost.)

(* * rht 11/14/86%: Now makes sure hung var PutSuccessfulLoc is NIL if we were passed an OverrideStream.)

(* * rht 7/1/87%: Now only turns off dirty flg if no OverrideStream.)

```
                                                (* Check to make sure this is an active note card.)
    (DECLARE (GLOBALVARS NC.LinksIdentifier))
    (AND (NEQ (fetch (Card Status) of Card)
              'ACTIVE)
         (NC.ReportError "NC.PutLinks" (CONCAT (NC.FetchTitle Card)
                                               " is not an active note card.")))
    (LET (PutSuccessfulLoc)
         (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
             (NC.DoCardPartFn Put Card 'LINKS (LET ((Stream (OR (STREAMP OverrideStream)
                                                                (NC.CoerceToNoteFileStream Card)))
                                                    StartLoc EndLoc)
```

(* * Write the links data at the end of the database file.)

```
                                                    (SETQ StartLoc (GETFILEPTR Stream))
                                                    (NC.WriteCardPartHeader Card NC.LinksIdentifier
                                                           [COND
                                                              (UseOldDateFlg (NC.FetchLinksDate Card))
                                                              (T (NC.SetLinksDate Card (DATE]
                                                           Stream)
                                                    (NC.WriteListOfLinks Stream (NC.FetchToLinks Card))
                                                    (NC.WriteListOfLinks Stream (NC.FetchFromLinks Card))
                                                    (NC.WriteListOfLinks Stream (NC.FetchGlobalLinks Card))
```

(* * Update the length field at the beginning of the card info)

```
                                                    (SETQ EndLoc (GETFILEPTR Stream))
                                                    (SETFILEPTR Stream StartLoc)
                                                    (NC.WritePtr Stream (DIFFERENCE EndLoc StartLoc)
                                                           3)
                                                    (SETFILEPTR Stream EndLoc)
```

(* * Now update the index to point to the link data just written. Done last in case writing of links doesn't complete okay.)

```
                                                    (OR OverrideStream (NC.SetLinksDirtyFlg Card))
```

(* * Now, since we were successful, we'll bind the PutSuccessfulLoc variable, which will be used freely by the AFTER PutCardPartFns to determine first if the Put succeeded, and if so, where in the stream it was Put.)

(* * Don't put a reasonable value in the hung variable PutSuccessfulLoc if we were passed OverrideStream.)

```
                                                    (SETQ PutSuccessfulLoc (if OverrideStream
                                                                               then NIL
                                                                            else StartLoc))
                                                    Card)))])
```

(NC.PutFromLinks

```
[LAMBDA (Card)                                                    ; Edited  3-Dec-87 18:59 by rht:
```

(* * The top level function for writing out only the FROMLINKS of a card.
Became a necessary function in the process of implementing the Server.)

(* * rht 9/11/86%: Now checks for card active via NC.CardActiveP rather than by checking for ACTIVE status.)

(* * rht 9/30/86%: Undid change of |9/11/86.| Card may not have ActiveFlg set when we're doing PutFromLinks.)

```
  (DECLARE (GLOBALVARS NC.LinksIdentifier))                       (* Check to make sure this is an active note card.)
  (if (NEQ (NC.FetchStatus Card)
           'ACTIVE)
      then (NC.ReportError "NC.PutFromLinks" (CONCAT (NC.FetchTitle Card)
                                                     " is not an active note card.")))
  (LET (PutSuccessfulLoc)
       (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
           (RESETLST
               (RESETSAVE (APPLY* (fetch (NoteFile PutCardPartFn) of (fetch (Card NoteFile) of Card))
                                  Card
                                  'FROMLINKS
                                  'BEFORE)
                          `(APPLY* ,(fetch (NoteFile PutCardPartFn) of (fetch (Card NoteFile) of Card))
                                   ,Card FROMLINKS AFTER))
               (LET ((Stream (NC.CoerceToNoteFileStream Card))
                     StartLoc)
```

(* * Write the links data at the end of the database file.)

```
                    (SETQ StartLoc (GETFILEPTR Stream))
                    (NC.WriteCardPartHeader Card NC.LinksIdentifier (NC.SetLinksDate Card (DATE))
                           Stream)
                    (NC.WriteListOfLinks Stream (NC.FetchFromLinks Card))
```

(* * Now update the index to point to the link data just written. Done last in case writing of links doesn't complete okay.)

```
                    (NC.SetLinksDirtyFlg Card)
```

(* * Now, since we were successful, we'll bind the PutSuccessfulLoc variable, which will be used freely by the AFTER PutCardPartFns to determine first if the Put succeeded, and if so, where in the stream it was Put.)

```
                    (SETQ PutSuccessfulLoc StartLoc)
                    Card)))])
```

(NC.PutRegion

```
[LAMBDA (Card)                                                    (* fgh%: "31-Aug-86 23:03")
```

(* * rht 1/31/85%: Now reads pointers from index array rather than file.)

(* * rht 11/12/85%: Updated to handle new Notefile and cardID format.)

(* * fgh |11/20/85| Added call to NC.ReadCardPartHeader)

(* * rht 1/23/86%: Changed to use NC.CoerceToNoteFileStream)

(* * kef 7/28/86%: Changed to play by multi-user device vector rules.)

(* * fgh |8/31/86| Adpated to use NC.DoCardPartFn.)

```
(LET (PutSuccessfulLoc)
     (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
         (NC.DoCardPartFn Put Card 'REGION
```

(* * Check to make sure this is an active note card.)

```
                (AND (NEQ (fetch (Card Status) of Card)
                          'ACTIVE)
                     (NC.ReportError "NC.PutRegion" (CONCAT (NC.FetchTitle Card)
                                                            " is not an active note card.")))
                (NC.WriteRegion Card (NC.CoerceToNoteFileStream Card))
                (SETQ PutSuccessfulLoc T)
                Card))])
```

(NC.PutTitle

```
[LAMBDA (Card UseOldDateFlg OverrideStream)                    (* rht%: " 1-Jul-87 10:23")
```

(* * Put the title of card ID onto DatabaseStream)

(* * rht 7/9/85%: Now puts out date after identifier. If UseOldDateFlg is non-nil, then use old date, otherwise use current date.)

(* * rht 11/10/85%: Updated to handle NoteFile and CardID scheme.)

(* * fgh |11/20/85| Added call to NC.WriteCardPartHeader and the mechanism to write the start and end pointers of the substance before calling the card type's putfn.)

(* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

(* * kef 7/16/86%: Makes use of the NoteFile device vector PutCardPartFn.)

(* * fgh |8/31/86| Adpated to use NC.DoCardPartFn.)

(* * pmi |11/4/86| Reinstated Randy's change (|1/23/86|) which somehow got lost.)

(* * rht 11/14/86%: Now makes sure hung var PutSuccessfulLoc is NIL if we were passed an OverrideStream.)

(* * rht 7/1/87%: Now only turns off dirty flg if no OverrideStream.)

```
                                                    (* Check to make sure this is an active note card.)
(DECLARE (GLOBALVARS NC.TitlesIdentifier))
(AND (NEQ (fetch (Card Status) of Card)
          'ACTIVE)
     (NC.ReportError "NC.PutTitle" (CONCAT (NC.FetchTitle Card)
                                           " is not an active note card.")))
(LET (PutSuccessfulLoc)
     (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
         (NC.DoCardPartFn Put Card 'TITLE (LET ((Stream (OR (STREAMP OverrideStream)
                                                            (NC.CoerceToNoteFileStream Card)))
                                                StartLoc EndLoc)
```

(* * First write out the title.)

```
                                                (SETQ StartLoc (GETFILEPTR Stream))
                                                (NC.WriteCardPartHeader Card NC.TitlesIdentifier
                                                       [COND
                                                          (UseOldDateFlg (NC.FetchTitleDate Card))
                                                          (T (NC.SetTitleDate Card (DATE]
                                                       Stream)
                                                (NC.WriteTitle Stream (NC.FetchTitle Card))
```

(* * Update the length field at the beginning of the card info)

```
                                                (SETQ EndLoc (GETFILEPTR Stream))
                                                (SETFILEPTR Stream StartLoc)
                                                (NC.WritePtr Stream (DIFFERENCE EndLoc StartLoc)
                                                       3)
                                                (SETFILEPTR Stream EndLoc)
```

(* * Now update the Index to reflect the new data just written. Done last in case the substance putting bombed for some reason.)

```
                                                (OR OverrideStream (NC.SetTitleDirtyFlg Card))
```

(* * Now, since we were successful, we'll bind the PutSuccessfulLoc variable, which will be used freely by the AFTER PutCardPartFns to determine first if the Put succeeded, and if so, where in the stream it was Put.)

(* * Don't put a reasonable value in the hung variable PutSuccessfulLoc if we were passed OverrideStream.)

```
                                                (SETQ PutSuccessfulLoc (if OverrideStream
```

```
                                                                    then NIL
                                                                   else StartLoc))
                                                Card)))])
```

(NC.PutPropList

```
[LAMBDA (Card UseOldDateFlg OverrideStream)                    (* rht%: " 1-Jul-87 10:23")
```

(* * Put the prop list for ID onto the database file.)

(* * rht 1/30/85%: Changed to use index array instead of file.)

(* * rht 7/9/85%: Now puts out date after identifier. If UseOldDateFlg is non-nil, then use old date, otherwise use current date.)

(* * rht 11/10/85%: Updated to handle NoteFile and Card scheme.)

(* * fgh |11/20/85| Added call to NC.WriteCardPartHeader and the mechanism to write the start and end pointers of the substance before calling the card type's putfn.)

(* * rht 1/23/86%: Now takes optional OverrideStream arg. This, if given, overrides stream of card's notefile.)

(* * kef 7/16/86%: Makes use of the NoteFile device vector PutCardPartFn.)

(* * fgh |8/31/86| Adapted to use NC.DoCardPartFn.)

(* * pmi |11/4/86| Reinstated Randy's change (|1/23/86|) which somehow got lost.)

(* * rht 11/14/86%: Now makes sure hung var PutSuccessfulLoc is NIL if we were passed an OverrideStream.)

(* * rht 7/1/87%: Now only turns off dirty flg if no OverrideStream.)

```
                                                    (* Check to make sure this is an active note card.)
  (DECLARE (GLOBALVARS NC.PropsIdentifier))
  (AND (NEQ (fetch (Card Status) of Card)
            'ACTIVE)
       (NC.ReportError "NC.PutPropList" (CONCAT (NC.FetchTitle Card)
                                                " is not an active note card.")))
  (LET (PutSuccessfulLoc)
       (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
           (NC.DoCardPartFn Put Card 'PROPLIST (LET ((Stream (OR (STREAMP OverrideStream)
                                                                 (NC.CoerceToNoteFileStream Card)))
                                                     StartLoc EndLoc)
```

(* * Write the proplist at the end of the database file.)

```
                                                     (SETQ StartLoc (GETFILEPTR Stream))
                                                     (NC.WriteCardPartHeader Card NC.PropsIdentifier
                                                            [COND
                                                               (UseOldDateFlg (NC.FetchPropListDate Card))
                                                               (T (NC.SetPropListDate Card (DATE]
                                                            Stream)
                                                     (NC.WritePropList Stream (NC.FetchPropList Card))
```

(* * Update the length field at the beginning of the card info)

```
                                                     (SETQ EndLoc (GETFILEPTR Stream))
                                                     (SETFILEPTR Stream StartLoc)
                                                     (NC.WritePtr Stream (DIFFERENCE EndLoc StartLoc)
                                                            3)
                                                     (SETFILEPTR Stream EndLoc)
```

(* * Now update the index to point to the proplist just written. Done last in case writing of proplist doesn't complete okay.)

```
                                                     (OR OverrideStream (NC.SetPropListDirtyFlg Card))
```

(* * Now, since we were successful, we'll bind the PutSuccessfulLoc variable, which will be used freely by the AFTER PutCardPartFns to determine first if the Put succeeded, and if so, where in the stream it was Put.)

(* * Don't put a reasonable value in the hung variable PutSuccessfulLoc if we were passed OverrideStream.)

```
                                                     (SETQ PutSuccessfulLoc (if OverrideStream
                                                                                then NIL
                                                                               else StartLoc))
                                                    Card)))])
```

(NC.PutNoteFileHeader

```
[LAMBDA (NoteFile)                                             (* rht%: "15-Nov-85 20:31")
```

(* * Write down to the notefile the header information extracted from the NoteFile object.)

```
  (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
        (FullFileName (fetch (NoteFile FullFileName) of NoteFile))
        (UID (fetch (NoteFile UID) of NoteFile))
        (NextIndexNum (fetch (NoteFile NextIndexNum) of NoteFile))
        (HashArraySize (fetch (NoteFile HashArraySize) of NoteFile))
        (NextLinkNum (fetch (NoteFile NextLinkNum) of NoteFile))
```

```
          (CheckptPtr (fetch (NoteFile CheckptPtr) of NoteFile))
          (Version (fetch (NoteFile Version) of NoteFile)))
         (WITH.MONITOR (NC.FetchMonitor NoteFile)
             (if (OPENP Stream)
                 then                                          (* Fill in the 30 information bytes for the notefile.)
                      (SETFILEPTR Stream 0)                    (* 3 bytes for next card ID)
                      (NC.WritePtr Stream NextIndexNum 3)      (* 3 bytes for index size)
                      (NC.WritePtr Stream HashArraySize 3)     (* One dummy byte so that version number stays in favorite old
                     place.)
                      (NC.WritePtr Stream -1 1)                (* 1 byte for notecards version number)
                      (NC.WritePtr Stream Version 1)           (* 3 bytes for next link ID)
                      (NC.WritePtr Stream NextLinkNum 3)       (* 3 bytes for pointer to current checkpt ptr.)
                      (NC.WritePtr Stream CheckptPtr 3)        (* 14 bytes for NoteFile UID.)
                      (NC.WriteUID NoteFile UID)               (* 1 bytes for future needs)
                      (NC.WritePtr Stream -1 1)
                      NoteFile
               else (NC.ReportError NIL "NC.PutNoteFileHeader: Stream not open!!!"))])
```

(NC.PutCheckptPtr

```
  [LAMBDA (NoteFile Ptr)                                       (* rht%: "15-Nov-85 00:31")

          (* * Write down a checkpoint pointer in the proper place on the notefile's header.
          Note that this changes the in-core value of the checkpoint ptr as well as on the file.)

    (WITH.MONITOR (NC.FetchMonitor NoteFile)
        (LET ((Stream (fetch (NoteFile Stream) of NoteFile)))
             (SETFILEPTR Stream 11)
             (NC.WritePtr Stream Ptr 3)
             (replace (NoteFile CheckptPtr) of NoteFile with Ptr)))])

)

;;; Functions for reading things off the notefile. Expect file pointer to already be set.

(DEFINEQ
```

(NC.ReadCardPartHeader

```
  [LAMBDA (Card Identifier Stream Don'tCheckForIDMismatchFlg)    (* rht%: "28-Jan-86 15:01")

          (* * Read the header for a card part and return the date from the header)

          (* * kirk 22Dec85 added NoteFile local)

          (* * rht 1/23/86%: Now takes optional stream arg. This, if given, overrides stream of card's notefile.)

          (* * rht 1/28/86%: Now takes Don'tCheckForIDMismatchFlg arg.
          If non-nil, then don't force UID on stream to match UID of card.)

    (OR (STREAMP Stream)
        (SETQ Stream (NC.CoerceToNoteFileStream Card)))
    (LET (VersionNumber Date ActualID CardUID)

          (* * Skip the length info)

         (NC.ReadPtr Stream 3)

          (* * Read the identifier and the version)

         (if (NOT (SETQ VersionNumber (NC.ReadIdentifier Stream Identifier)))
             then (NC.ReportError "NC.ReadCardPartHeader" (CONCAT (NC.FetchTitle Card)
                                                                  " Error while reading NoteFile" " -- incorrect
                                                                  identifier.")))
         (if (GEQ VersionNumber 1)
             then (SETQ Date (NC.ReadDate Stream)))
         (SETQ ActualID (NC.ReadUID Stream))
         (if [AND (NOT Don'tCheckForIDMismatchFlg)
                  (NOT (NC.SameUIDP ActualID (SETQ CardUID (fetch (Card UID) of Card]
             then (NC.ReportError "NC.ReadCardPartHeader" (CONCAT "ID mismatch while reading item.  Expected ID:
" CardUID "   Found ID: " ActualID)))
        Date])
```

(NC.ReadIdentifier

```
  [LAMBDA (NoteFileOrStream Identifier)                        (* rht%: " 1-Nov-86 15:10")
                                                               (* Return T if next item on databaseStream is the identifier
                                                               specified by Identifier)

          (* * rht 2/4/85%: A horrible hack for the case of titles identifier. This is because a previous typo was causing NOBIND to get
          written for titles identifiers.)

          (* * rht 7/9/85%: Now checks for new data format. This is indicated by identifiers with the last two %#'s clipped off.
          Then comes the one-byte version number of the data format. If identifier is not clipped then it's old style and there is no
          version number. Return version number if there is one, 0 if old style, and NIL if can't match identifier.)

          (* * fkr |11/8/85| Changed to handle NoteFile object.)
```

```
          (* * rht 11/24/85%: No longer worries about the screwy NOBIND Titles identifier.
          Also assumes Identifier is already clipped.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream)))
         (if [AND (EQ (READ Stream NC.OrigReadTable)
                      Identifier)
                  (NUMBERP (PROGN                                    (* First char is separator. Next is one-byte version number.)
                                  (BIN Stream)
                                  (NC.ReadPtr Stream 1]
          else NIL])
```

(NC.ReadRegion

```
  [LAMBDA (NoteFileOrStream)                                         (* rht%: "23-Jan-86 18:16")
```

```
          (* * fkr 11/885%: Now takes NoteFile arg. No more ID arg.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream)))
         (CREATEREGION (NC.ReadPtr Stream 2)
                (NC.ReadPtr Stream 2)
                (NC.ReadPtr Stream 2)
                (NC.ReadPtr Stream 2])
```

(NC.ReadListOfLinks

```
  [LAMBDA (NoteFileOrStream)                                         (* rht%: " 1-Nov-86 15:09")
```

```
          (* * Read a list of link records from the notefile. Create a datatype instance for each.)

          (* * rht 11/14/85%: Now uses NC.CardFromUID to get Card object from hash array.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

          (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream)))
         (PROG1 (bind Link while (SETQ Link (NC.ReadLink Stream)) collect Link)

          (* * Read past the end of list identifier)

             (READ Stream NC.OrigReadTable)
             (BIN Stream))])
```

(NC.ReadUID

```
  [LAMBDA (NoteFileOrStream LookUpInHashArrayFlg)                    ; Edited 11-May-88 21:08 by Trigg
```

;;; Get a UID off of the file. Since UIDs are BIGNUMs less than {EXPT 2 112} just read their 112 bits from the file.

;; rht 5/11/88: Takes new arg LookUpInHashArrayFlg. If non-nil, then call NC.LookUpUIDInHashArray to return existing equivalent UID object if ;; there is one.

```
    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream))
          UID)
         (SETQ UID (create UID
                           UID0 _ (BIN16 Stream)
                           UID1 _ (BIN16 Stream)
                           UID2 _ (BIN16 Stream)
                           UID3 _ (BIN16 Stream)
                           UID4 _ (BIN16 Stream)
                           UID5 _ (BIN16 Stream)
                           UID6 _ (BIN16 Stream)))
         (PROG1 (if LookUpInHashArrayFlg
                    then (NC.LookUpUIDInHashArray UID)
                  else UID)
```

;;; skip past CR following UID

```
             (BIN Stream))])
```

(NC.ReadDate

```
  [LAMBDA (NoteFileOrStream)                                         (* pmi%: " 6-Apr-87 12:52")
```

```
          (* * Read a date string from Stream. All dates have the same length, so can use that as a check.
          I'm allowing null date since we may be compacting an old style
          (non-dated) notefile. Thus we won't give it a misleadingly new date.)

          (* * rht 11/11/85%: Now handles new notefile object.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)
```

```
          (* * rht 11/1/86%: Now uses our readtable when reading.)

          (* * pmi 4/6/87%: Removed call to NC.ReportError if bad date was found;
          instead, just return NIL.)

    (DECLARE (GLOBALVARS NC.OrigReadTable NC.DateStringLength))
    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream))
          Date)

          (* * Read Date)

         (SETQ Date (READ Stream NC.OrigReadTable))

          (* * Read past CR)

         (BIN Stream)
         (COND
            ((OR (NULL Date)
                 (EQ (NCHARS Date)
                     NC.DateStringLength))
             Date)
            (T

          (* * (NC.ReportError "NC.ReadDate" (CONCAT Date " is not a proper date.")))

               NIL])
```

(NC.ReadCardType

```
  [LAMBDA (NoteFileOrStream)                                    (* rht%: " 1-Nov-86 15:11")

          (* * Get a card type off of the file.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

          (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream)))
         (PROG1 (READ Stream NC.OrigReadTable)

          (* * read past CR)

            (BIN Stream))])
```

(NC.ReadTitle

```
  [LAMBDA (NoteFileOrStream)                                    (* rht%: " 1-Nov-86 15:11")

          (* * Get a title off of the file.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

          (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (READ (NC.CoerceToNoteFileStream NoteFileOrStream)
          NC.OrigReadTable])
```

(NC.ReadPropList

```
  [LAMBDA (NoteFileOrStream)                                    (* rht%: " 1-Nov-86 15:12")

          (* * Get a prop list off of the file.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

          (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (READ (NC.CoerceToNoteFileStream NoteFileOrStream)
          NC.OrigReadTable])
```

(NC.ReadLink

```
  [LAMBDA (Stream)                                              ; Edited 15-Jul-2021 21:46 by frank

          (* * Read a single Link DATAYPE instance from Stream)

          (* * Link identifier and CR)

          (* * rht 11/25/85%: Now handles case of version 2 style link.)

          (* * fgh |8/27/86| Put in Kludge to handle problem with some versions of WRITE.IMAGEOBJ that put in an extra CR before
          the stuff.)
```

```
          (* * rht 11/1/86%: Now uses our readtable when reading.)

          (* * fgh #.(SEDIT::MAKE-BROKEN-ATOM "7/15/21:") Changed CR KLUDGE to include CR or LF)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((FirstChar (NC.ReadPtr Stream 1)))
         (COND
            ((ZEROP FirstChar)

          (* * Version Number 0 means end of list of links)

             NIL)
            ((EQ FirstChar 1)

          (* * The link info for a version 3 style link.)

             (PROG1 (create Link
                            UID _ (NC.ReadUID Stream)
                            SourceCard _ (NC.CardOrCardHolderFromUID (NC.ReadUID Stream)
                                                 (NC.ReadUID Stream))
                            DestinationCard _ (NC.CardOrCardHolderFromUID (NC.ReadUID Stream)
                                                      (NC.ReadUID Stream))
                            AnchorMode _ (READ Stream NC.OrigReadTable)
                            Label _ (READ Stream NC.OrigReadTable)
                            DisplayMode _ (READ Stream NC.OrigReadTable))

          (* * Get that damn CR)

                    (BIN Stream)))
            ((OR (EQ FirstChar 13)
                 (EQ FirstChar 10))

          (* * KLUDGE to account for fact that some versions of WRITE.IMAGEOBJ put a CR or a LF before beginning the actual
          stuff.)

             (NC.ReadLink Stream))
            (T

          (* * Version 2 style link. Move back over the left parantheses and read the link as a list.)

               (SETFILEPTR Stream (SUB1 (GETFILEPTR Stream)))
               (READ Stream NC.OrigReadTable])

)


;;; Functions for writing things on the notefile. Expect file pointer to already be set.

(DEFINEQ

(NC.WriteCardPartHeader
  [LAMBDA (Card Identifier Date Stream)                    (* rht%: "23-Jan-86 17:50")

          (* * write the header of a card part onto the NoteFile)

          (* * rht 1/23/86%: Now takes optional stream arg. This, if given, overrides stream of card's notefile.)

    (OR (STREAMP Stream)
        (SETQ Stream (NC.CoerceToNoteFileStream Card)))

          (* * leave space for the length information)

    (NC.WritePtr Stream 0 3)

          (* * write the card part identifier)

    (NC.WriteIdentifier Stream Identifier)

          (* * write the date)

    (NC.WriteDate Stream Date)

          (* * write the cards uid)

    (NC.WriteUID Stream (fetch (Card UID) of Card])

(NC.WriteIdentifier
  [LAMBDA (NoteFileOrStream Identifier)                    (* rht%: " 1-Nov-86 15:05")
                                                           (* Put Identifier on DatabaseStream)

          (* * Now puts out new data format style. This consists of the identifier with the last two %#'s clipped off followed by the data
          format version byte.)

          (* * rht 11/12/85%: Now handles new notefile format.)

          (* * rht 11/24/85%: Assumes identifier is already clipped.)
```

```
          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

    (DECLARE (GLOBALVARS NC.DataFormatVersionNumber NC.OrigReadTable))
    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream)))
         (PRINT Identifier Stream NC.OrigReadTable)
         (NC.WritePtr Stream NC.DataFormatVersionNumber 1])
```

(NC.WriteRegion

```
  [LAMBDA (Card Stream)                                    ; Edited 27-Jul-90 09:56 by tafel

          (* * rht 10/3/85%: Now checks first for a saved region (probably there because card got moved off screen)%.)

          (* * fkr 11/8/85%: Changed to handle CardIDs and NoteFile objects.)

          (* * rht 1/23/86%: Now takes optional stream arg. This, if given, overrides stream of card's notefile.)

          (* * rht 7/5/86%: Now makes sure region fits on screen.)

    (OR (STREAMP Stream)
        (SETQ Stream (NC.CoerceToNoteFileStream Card)))
    (LET (Window Region)
         (SETQ Window (NC.FetchWindow Card))
         [SETQ Region (MAKEWITHINREGION (create REGION copying (OR (NC.FetchSavedRegion Card)
                                                                   (AND Window (WINDOWPROP Window 'REGION))
                                                                   (NC.FetchRegion Card)
                                                                   (NC.MakeDummyRegion Card]
         (AND (NC.ActiveCardP Card)
              (NC.SetRegion Card Region))
         (NC.WritePtr Stream (fetch (REGION LEFT) of Region)
                2)
         (NC.WritePtr Stream (fetch (REGION BOTTOM) of Region)
                2)
         (NC.WritePtr Stream (fetch (REGION WIDTH) of Region)
                2)
         (NC.WritePtr Stream (fetch (REGION HEIGHT) of Region)
                2])
```

(NC.WriteListOfLinks

```
  [LAMBDA (NoteFileOrStream Links)                         (* rht%: " 1-Nov-86 15:09")

          (* * Write the given links down to the notefile, coercing NoteFile object to NF-UID if necessary.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)

          (* * rht 11/1/86%: Now uses our readtable when printing.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream)))
         (for Link in Links when (type? Link Link) do (NC.WriteLink Link Stream))
         (NC.WritePtr Stream 0 1)
         (PRINT 'EndLinks Stream NC.OrigReadTable])
```

(NC.WriteUID

```
  [LAMBDA (NoteFileOrStream UID)                           (* rht%: "23-Jan-86 17:44")

          (* * Write a UID out to Stream. Since UIDs are BIGNUMs less than {EXPT 2 112} just write their 112 bits to the file.)

          (* * rht 11/12/85%: Handles new notefile format.)

    (LET ((Stream (NC.CoerceToNoteFileStream NoteFileOrStream)))
         (AND (type? UID UID)
              (BOUT16 Stream (ffetch (UID UID0) of UID))
              (BOUT16 Stream (ffetch (UID UID1) of UID))
              (BOUT16 Stream (ffetch (UID UID2) of UID))
              (BOUT16 Stream (ffetch (UID UID3) of UID))
              (BOUT16 Stream (ffetch (UID UID4) of UID))
              (BOUT16 Stream (ffetch (UID UID5) of UID))
              (BOUT16 Stream (ffetch (UID UID6) of UID)))

          (* * End with a CR)

         (BOUT Stream 13])
```

(NC.WriteDate

```
  [LAMBDA (NoteFileOrStream Date)                          (* rht%: " 1-Nov-86 15:05")

          (* * Write a date string out to Stream.)

          (* * rht 11/12/85%: Handles new notefile format.)

          (* * rht 1/23/86%: Now takes notefile or stream as arg.)
```

(* * rht 11/1/86%: Now uses our readtable when printing.)

```
    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PRINT Date (NC.CoerceToNoteFileStream NoteFileOrStream)
           NC.OrigReadTable])
```

(NC.WriteCardType

```
  [LAMBDA (NoteFileOrStream CardType)                    (* rht%: " 1-Nov-86 15:05")
```

(* * Writes a card type down to notefile.)

(* * rht 1/23/86%: Now takes notefile or stream as arg.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

```
    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PRINT CardType (NC.CoerceToNoteFileStream NoteFileOrStream)
           NC.OrigReadTable])
```

(NC.WriteTitle

```
  [LAMBDA (NoteFileOrStream Title)                       (* rht%: " 1-Nov-86 15:06")
```

(* * Write a title out to Stream.)

(* * rht 1/23/86%: Now takes notefile or stream as arg.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

```
    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PRINT Title (NC.CoerceToNoteFileStream NoteFileOrStream)
           NC.OrigReadTable])
```

(NC.WritePropList

```
  [LAMBDA (NoteFileOrStream PropList)                    (* rht%: " 1-Nov-86 15:06")
```

(* * Write a prop list out to Stream.)

(* * rht 1/23/86%: Now takes notefile or stream as arg.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

```
    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PRINT PropList (NC.CoerceToNoteFileStream NoteFileOrStream)
           NC.OrigReadTable])
```

(NC.WriteLink

```
  [LAMBDA (Link Stream)                                  (* rht%: " 1-Nov-86 15:06")
```

(* * Write a single link DATAYPE instance down to Stream)

(* * rht 11/1/86%: Now uses our readtable when printing.)

```
    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((SourceCard (fetch (Link SourceCard) of Link))
          (DestinationCard (fetch (Link DestinationCard) of Link)))
```

(* * This is version 1 link format)

```
         (NC.WritePtr Stream 1 1)
         (NC.WriteUID Stream (fetch (Link UID) of Link))
         (NC.WriteUID Stream (fetch (Card UID) of SourceCard))
         (NC.WriteUID Stream (fetch (NoteFile UID) of (fetch (Card NoteFile) of SourceCard)))
         (NC.WriteUID Stream (fetch (Card UID) of DestinationCard))
         (NC.WriteUID Stream (fetch (NoteFile UID) of (fetch (Card NoteFile) of DestinationCard)))
         (PRINT (fetch (Link AnchorMode) of Link)
                Stream NC.OrigReadTable)
         (PRINT (fetch (Link Label) of Link)
                Stream NC.OrigReadTable)
         (PRINT (fetch (Link DisplayMode) of Link)
                Stream NC.OrigReadTable])

)

;;; The NoteFile object, Notefiles hash array and accompanying functions.

(DEFINEQ

## (NC.NoteFileFromNoteFileUID

  [LAMBDA (NoteFileUID)                                  (* rht%: "14-Nov-85 23:36")
```

(* * Return the NoteFile datatype instance corresponding to the NFUID by looking it up in the notefiles hash array.)

(* * rht 11/14/85%: This should get smarter in the future when have cross-file linking.
There it should check some file containing path name hints if it can't find UID in hash table.)

```
(GETHASH NoteFileUID NC.NoteFilesHashArray])
```

(NC.FetchTopLevelCards

```
[LAMBDA (NoteFile)                                          (* rht%: "15-Nov-85 01:59")
```

(* * Return a list of the three top level fileboxes.)

```
(LIST (fetch (NoteFile TableOfContentsCard) of NoteFile)
      (fetch (NoteFile OrphansCard) of NoteFile)
      (fetch (NoteFile ToBeFiledCard) of NoteFile])
```

(NC.StoreNoteFileInHashArray

```
[LAMBDA (NoteFileOrFileName)                                ; Edited  8-Dec-88 16:24 by krivacic
```

;;; store NoteFile in HashArray, a NoteFiles hash array.

;;; fgh 5/23/86 First created.

;;; Are there any other NoteFiles of the same fullfilename here? If so, something got messed up. Remove these items.

;;; pmi 5/19/87: Removed HashArray argument --- this function only applies to NC.NoteFilesHashArray.

```
(DECLARE (GLOBALVARS NC.NoteFilesHashArray))
(LET [(NoteFile (if (type? NoteFile NoteFileOrFileName)
                    then NoteFileOrFileName
                  else (NC.NoteFileFromFileName NoteFileOrFileName]
     (if NoteFile
         then [MAPHASH NC.NoteFilesHashArray (FUNCTION (LAMBDA (NF UID)
                                                 (if (EQ (fetch (NoteFile FullFileName) of NoteFile)
                                                         (fetch (NoteFile FullFileName) of NF))
                                                     then (PUTHASH UID NIL NC.NoteFilesHashArray]
```

(* * Okay, now put in the new entry)

```
            (PUTHASH (fetch (NoteFile UID) of NoteFile)
                   NoteFile NC.NoteFilesHashArray])
```

(NC.RemoveNoteFile

```
[LAMBDA (NoteFileOrFileNameOrUID)                           (* pmi%: "28-May-87 11:54")
```

(* * Remove this notefile from the global hash array and the list of noticed notefiles.)

(* * pmi 5/28/87%: Checks more thoroughly for type of object passed in.
Also calls NC.RemoveNoteFileName to remove NoteFile name form list of Noticed NoteFiles.
Returns NoteFile object removed.)

```
(DECLARE (GLOBALVARS NC.NoteFilesHashArray))
(LET (NoteFile NoteFileName)
     (if (type? NoteFile NoteFileOrFileNameOrUID)
         then (SETQ NoteFile NoteFileOrFileNameOrUID)
              (SETQ NoteFileName (fetch (NoteFile FullFileName) of NoteFile))
       elseif (type? UID NoteFileOrFileNameOrUID)
         then (SETQ NoteFile (GETHASH NoteFileOrFileNameOrUID NC.NoteFilesHashArray))
              (SETQ NoteFileName (fetch (NoteFile FullFileName) of NoteFile))
       else (SETQ NoteFile (NC.NoteFileFromFileName NoteFileOrFileNameOrUID))
            (SETQ NoteFileName NoteFileOrFileNameOrUID))
     (if NoteFile
         then (NC.RemoveNoteFileFromHashArray NoteFile))
     (if NoteFileName
         then (NC.RemoveNoteFileName NoteFileName))
    NoteFile])
```

(NC.TotalIndexSize

```
[LAMBDA (HashArraySize)                                     (* rht%: " 8-Nov-85 13:19")
```

(* * Return the length of the index part of the notefile including header.)

```
(PLUS (CONSTANT (fetch (NoteFileVersion NoteFileHeaderSize) of (NC.FetchCurrentVersionObject)))
      (TIMES (CONSTANT (fetch (NoteFileVersion NoteFileIndexWidth) of (NC.FetchCurrentVersionObject)))
             HashArraySize])
```

(NC.NoteFileLocFromIndexNum

```
[LAMBDA (IndexNum)                                          (* rht%: "14-Nov-85 22:45")
```

(* * Compute the location on the file of the index entry for IndexNum.)

```
(PLUS (CONSTANT (fetch (NoteFileVersion NoteFileHeaderSize) of (NC.FetchCurrentVersionObject)))
      (TIMES (CONSTANT (fetch (NoteFileVersion NoteFileIndexWidth) of (NC.FetchCurrentVersionObject)))
             (SUB1 IndexNum])
```

(NC.FetchMonitor

```
  [LAMBDA (NoteFile)                                    ; Edited 20-May-2021 18:11 by pi

          (* * fetch the monitor lock from a NoteFile and make sure that it is okay.)

    (LET ((MonitorLock (fetch (NoteFile MonitorLock) of NoteFile)))
         (if (EQ (TYPENAME MonitorLock)
                 'MONITORLOCK)
             then MonitorLock
          else (NC.ReportError "NC.FetchMonitor" (CONCAT "No monitorlock on NoteFile -- " (fetch (NoteFile
                                                                                                  FullFileName
                                                                                                  )
                                                                                        of NoteFile])
```

(NC.SetMonitor

```
  [LAMBDA (NoteFile MonitorLock)                        (* rht%: "13-Nov-85 00:49")

          (* * rht 11/12/85%: Updated to use new NoteFile format.)

    (replace (NoteFile MonitorLock) of NoteFile with MonitorLock])
```

(NC.SameNoteFileP

```
  [LAMBDA (NF1 NF2)                                     (* fgh%: "16-Nov-85 00:27")

          (* * Are NF1 and NF2 the same NoteFile?)

    (EQ NF1 NF2])
```

(NC.ListOfNoteFiles

```
  [LAMBDA NIL                                           (* pmi%: "22-May-87 11:31")

          (* * Create a list of all NoteFiles found in the notefiles hash array.)

          (* * pmi 5/22/87%: Added Globalvars.)

    (DECLARE (GLOBALVARS NC.NoteFilesHashArray))
    (LET (NoteFiles)
         [MAPHASH NC.NoteFilesHashArray (FUNCTION (LAMBDA (Value Key)
                                                    (AND (type? NoteFile Value)
                                                         (push NoteFiles Value]
        NoteFiles])
```

(NC.NoteFileFromFileName

```
  [LAMBDA (FileName)                                    ; Edited 13-Dec-88 16:38 by krivacic
```

;;; Return the notefile object for the given file name or NIL if none. Done by checking notefiles hash array.

;;; rht 5/6/86: Now doesn't just return the first notefile object having the desired name. Tries to return one that's open if there are any.

;;; pmi 12/4/86: Probably not the best fix, but changed FullFileName to be (OR (FULLNAME FileName) FileName) (\, since) by the time we get here
;;; when deleting a file, the file is already gone and FULLNAME returns NIL

```
    (LET* ((FullFileName (OR (FULLNAME FileName)
                             FileName))
           (NoteFiles (for NoteFile in (NC.ListOfNoteFiles) when (STRING-EQUAL FullFileName (fetch (NoteFile
                                                                                                    FullFileName
                                                                                                    )
                                                                                          of NoteFile))
                             collect NoteFile)))
          (OR (for NoteFile in NoteFiles when (LET ((Stream (fetch (NoteFile Stream) of NoteFile)))
                                                (AND (STREAMP Stream)
                                                     (OPENP Stream)))
                 do (RETURN NoteFile))
              (CAR NoteFiles])

)

(DEFINEQ
```

(NC.RemoveNoteFileFromHashArray

```
  [LAMBDA (NoteFile)                                    (* pmi%: "21-May-87 11:16")

          (* * Remove this notefile from the global hash array.)

          (* * pmi 5/21/87%: Service function called by NC.RemoveNoteFile.)

    (DECLARE (GLOBALVARS NC.NoteFilesHashArray))
    (LET ((NoteFileUID (fetch (NoteFile UID) of NoteFile)))
         (if (AND NoteFile NoteFileUID)
             then (PUTHASH NoteFileUID NIL NC.NoteFilesHashArray])
```

```
(NC.RemoveNoteFileName
  [LAMBDA (NoteFileOrFileName)                                    (* pmi%: "18-Dec-87 10:34")

          (* * pmi 5/19/87%: Created to keep track of noticed NoteFiles)

          (* * pmi 8/13/87%: Added trashing of NC.NoticedNoteFilesMenu to force its recomputation.)

          (* * pmi 12/18/87%: Changed the global var NC.NoticedNoteFileNames to NCP.NoticedNoteFileNames to make it available
          in the programmer's interface.)

    (DECLARE (GLOBALVARS NCP.NoticedNoteFileNames NC.NoticedNoteFilesMenu))
    (LET (FullFileName)
         (if (type? NoteFile NoteFileOrFileName)
             then (SETQ FullFileName (fetch (NoteFile FullFileName) of NoteFileOrFileName))
           else (SETQ FullFileName (OR (FULLNAME NoteFileOrFileName)
                                       NoteFileOrFileName)))

          (* * If the filename is a valid or invalid file, remove it from the list of noticed files.
          If the resulting list is empty, set it to NIL (DREMOVE can't set a list to NIL))

         (if (DREMOVE FullFileName NCP.NoticedNoteFileNames)
           else (SETQ NCP.NoticedNoteFileNames NIL))

          (* * Trash the menu of noticed notefiles so that it will be recomputed.)

         (SETQ NC.NoticedNoteFilesMenu NIL])

(NC.NoticeNoteFile
  [LAMBDA (NoteFileOrFileName)                                    ; Edited  8-Dec-88 16:22 by krivacic

;;; store NoteFile in HashArray, a NoteFiles hash array.

;;; fgh 5/23/86 First created.

;;; pmi 5/27/87: Broke into two functions: one to add the filename to list of noticed files, and one to put the notefile in the notefile hash array.

    (NC.NoticeNoteFileName NoteFileOrFileName)
    (NC.StoreNoteFileInHashArray NoteFileOrFileName])

(NC.NoticeNoteFileName
  [LAMBDA (NoteFileOrFileName)                                    ; Edited 20-Dec-88 16:16 by pmi
    ;; pmi 5/14/87: Created to keep track of noticed NoteFiles
    ;; pmi 5/21/87: Now creates a menu item bitmap for this notefile.
    ;; pmi 8/13/87: Overhauled stuff for menu of noticed notefiles.
    ;; pmi 12/18/87: Changed the global var NC.NoticedNoteFileNames to NCP.NoticedNoteFileNames to make it available in the programmer's
    ;; interface.
    ;; rar.  10/20/88 Added check to see if Notefile is marked as "UnNoticable" and shouldn't appear in the menu.
    ;; pmi 12/20/88: Added check for non-NIL notefile before checking for UnNoticable property.
    (DECLARE (GLOBALVARS NCP.NoticedNoteFileNames NC.NoticedNoteFilesMenu))
    (LET (FullFileName NoteFile)
         (if (type? NoteFile NoteFileOrFileName)
             then (SETQ FullFileName (fetch (NoteFile FullFileName) of NoteFileOrFileName))
                  (SETQ NoteFile NoteFileOrFileName)
           else (SETQ FullFileName (FULLNAME NoteFileOrFileName))
                (SETQ NoteFile (NC.NoteFileFromFileName FullFileName)))

;;; If the filename is a valid file, add it to the list of noticed files.  If the filename is not a valid file, remove it from the list of noticed files.  If the resulting list
;;; is empty, set it to NIL (DREMOVE can't set a list to NIL)

         (if FullFileName
             then ;; If the notefile has not been marked as "UnNoticable", then add it to the menu
                  (if [NOT (AND NoteFile (NCP.NoteFileProp NoteFile 'UnNoticable]
                      then (if NCP.NoticedNoteFileNames
                               then (MERGEINSERT FullFileName NCP.NoticedNoteFileNames T)
                             else (SETQ NCP.NoticedNoteFileNames (MERGEINSERT FullFileName
                                                                              NCP.NoticedNoteFileNames T)))
                           (SELECTQ (GETPROP FullFileName 'LastKnownStatus)
                                (OPEN (if (NULL (NC.NoteFileOpenP NoteFile))
                                          then (PUTPROP FullFileName 'LastKnownStatus 'CLOSED)
                                               (SETQ NC.NoticedNoteFilesMenu NIL)))
                                (CLOSED (if (NC.NoteFileOpenP NoteFile)
                                            then (PUTPROP FullFileName 'LastKnownStatus 'OPEN)
                                                 (SETQ NC.NoticedNoteFilesMenu NIL)))
                                (PROGN (if (NC.NoteFileOpenP NoteFile)
                                           then (PUTPROP FullFileName 'LastKnownStatus 'OPEN)
                                         else (PUTPROP FullFileName 'LastKnownStatus 'CLOSED))
                                       (SETQ NC.NoticedNoteFilesMenu NIL)))

;;; Constuct menu items for this notefile.
```

```
                    (NC.CreateNoteFileMenuItems FullFileName))
          else (if (DREMOVE NoteFileOrFileName NCP.NoticedNoteFileNames)
                   else (SETQ NCP.NoticedNoteFileNames NIL])
```

(NC.NoticedNoteFileNamesMenu

```
  [LAMBDA (IncludeNewNoteFileFlg AllowedOperations InterestedWindow Operation)
                                                                    ; Edited 13-Sep-88 09:57 by pmi
```

;;; Bring up a menu of all notefiles found in the notefiles hash array. Also allow user to open a new notefile.

;;; kirk 23Jan86 Added AskYesOrNo and InterestedWindow parameter

;;; fgh 6/8/86 Added check to make sure NoteFile is open if it has a menu on the screen. Needed to handle case of liongering NF menus.

;;; fgh 6/24/86 Changed to be a general function rather than one specific for opening. Now just returns the chosen name. Also, added
;;; IncludeNewNoteFileFlg and ShowOnlyOpenNFsFlg. Removed InterestedWindow arg.

;;; fgh 6/27/86 Added InterestedWindow & Operation args and call to NC.DatabaseFileName.

;;; pmi 12/4/86: Added version numbers to rootnames on list of known files. Also cleaned up help string for menu items. It was giving a bogus message
;;; about opening the selected file, even though this function is used for many operations and not just for Open.

;;; pmi 2/18/87: Added GLOBALVARS declaration for NC.MenuFont

;;; pmi 5/15/87: Used to be NC.ListOfNoteFilesMenu. Changed symbol for open notefile to o. Now uses NCP.NoticedNoteFileNames instead of hash
;;; array to build menu. Returns a NoteFile name instead of a NoteFile object.

;;; pmi 5/21/87: Replaced each NoteFile menu item with a bitmap of its name in a large font and its full filename in a small font.

;;; pmi 8/20/87: Made modifications to speed up this menu: cache it when possible, only recompute the shading, etc.

;;; pmi 12/8/87: Cleaned up some of the shading; converted AllowedOperations to be one of Open, CLOSED or NIL for both.

;;; pmi 12/30/87: Changed the global var NC.NoticedNoteFileNames to NCP.NoticedNoteFileNames to make it available in the programmer's interface.
;;; Also wrapped U-CASE around all SELECTQ vars so that case doesn't matter.

;;; pmi 9/13/88: Changed to use one of 4 bitmaps for each menu item, depending on whether the notefile is open or closed, and whether an open or
;;; closed operation is in progress. Also now only puts "--Other Notefile--" on the menu if it makes sense to specify one (IncludeNewNoteFileFlg is
;;; non-NIL).

```
    (DECLARE (GLOBALVARS NC.FileNameFont NCP.NoticedNoteFileNames NC.NoticedNoteFilesMenu
                    NC.NoticedNoteFilesMenuNewItem WHITESHADE NCP.GrayShade))
    (LET (Result)
         [SETQ Result
          (PROG (Items)
           ;; Shade either the open or closed files, depending on the type allowed by this operation.
                [SETQ Items
                 (SELECTQ (U-CASE AllowedOperations)
                      (OPEN `[,@(for NoteFileName in NCP.NoticedNoteFileNames bind NoteFile
                                   collect (SELECTQ (U-CASE (GETPROP NoteFileName 'LastKnownStatus))
                                                    (OPEN (GETPROP NoteFileName 'OpenMenuItem))
                                                    (CLOSED (GETPROP NoteFileName 'ClosedMenuItemShaded))
                                                    NIL))
                             ,@(if IncludeNewNoteFileFlg
                                   then (if NC.NoticedNoteFilesMenuNewItem
                                            then (LIST NC.NoticedNoteFilesMenuNewItem)
                                          else (LIST (SETQ NC.NoticedNoteFilesMenuNewItem
                                                           '("-- Other NoteFile --" 'NEW "Select some other notefile -
                                                                you'll be prompted for the name."])
                      (CLOSED `[,@(for NoteFileName in NCP.NoticedNoteFileNames bind NoteFile
                                     collect (SELECTQ (U-CASE (GETPROP NoteFileName 'LastKnownStatus))
                                                      (OPEN (GETPROP NoteFileName 'OpenMenuItemShaded))
                                                      (CLOSED (GETPROP NoteFileName 'ClosedMenuItem))
                                                      NIL))
                               ,@(if IncludeNewNoteFileFlg
                                     then (if NC.NoticedNoteFilesMenuNewItem
                                              then (LIST NC.NoticedNoteFilesMenuNewItem)
                                            else (LIST (SETQ NC.NoticedNoteFilesMenuNewItem
                                                             '("-- Other NoteFile --" 'NEW "Select some other notefile
                                                                  - you'll be prompted for the name."])
                      `(,@(for NoteFileName in NCP.NoticedNoteFileNames bind NoteFile
                             collect (SELECTQ (U-CASE (GETPROP NoteFileName 'LastKnownStatus))
                                              (OPEN (GETPROP NoteFileName 'OpenMenuItem))
                                              (CLOSED (GETPROP NoteFileName 'ClosedMenuItem))
                                              NIL))
                        ,@(if IncludeNewNoteFileFlg
                              then (if NC.NoticedNoteFilesMenuNewItem
                                       then (LIST NC.NoticedNoteFilesMenuNewItem)
                                     else (LIST (SETQ NC.NoticedNoteFilesMenuNewItem '("-- Other NoteFile --"
                                                                                       'NEW "Select some other
                                                                                       notefile - you'll be
                                                                                       prompted for the name."]
                [if (NULL Items)
                    then (SELECTQ (U-CASE AllowedOperations)
                                  (OPEN (NC.PrintMsg InterestedWindow NIL "No open NoteFiles." (CHARACTER 13)))
```

```
                              (CLOSED (NC.PrintMsg InterestedWindow NIL "No closed NoteFiles." (CHARACTER 13)))
                              (NC.PrintMsg InterestedWindow NIL "No NoteFiles." (CHARACTER 13)))
                         (RETURN NIL)
                   elseif [AND (EQ (LENGTH Items)
                                   1)
                               (EQUAL (CADAR Items)
                                      ''NEW]
                     then (RETURN 'NEW)
                   else (SETQ NC.NoticedNoteFilesMenu
                         (create MENU
                                 ITEMS _ Items
                                 TITLE _ "NoteFiles"
                                 MENUFONT _ NC.FileNameFont
                                 ITEMHEIGHT _ (IPLUS (BITMAPHEIGHT (CAAR Items))
                                                     1]
                  (replace (MENU IMAGE) of NC.NoticedNoteFilesMenu with NIL)
                  (RETURN (MENU NC.NoticedNoteFilesMenu]
            (if (EQ Result 'NEW)
                then (SETQ Result (NC.DatabaseFileName (CONCAT "Name of NoteFile to " (SUBSTRING Operation 1 -9)
                                                               (CHARACTER 13))
                                                       " -- " T T NIL InterestedWindow)))
           Result])
```

(NC.NoteFileNoticedP

```
  [LAMBDA (NoteFileOrFileName)                                        (* pmi%: "18-Dec-87 10:17")
```

(* * pmi 6/2/87%: Created to check if notefile has been noticed by NoteCards.)

(* * pmi 12/18/87%: Changed the global var NC.NoticedNoteFileNames to NCP.NoticedNoteFileNames to make it available in the programmer's interface.)

```
    (DECLARE (GLOBALVARS NCP.NoticedNoteFileNames))
    (LET (FullFileName NoteFile)
         (if (type? NoteFile NoteFileOrFileName)
             then (SETQ NoteFile NoteFileOrFileName)
                  (SETQ FullFileName (fetch (NoteFile Menu) of NoteFile))
           elseif (SETQ FullFileName (FULLNAME NoteFileOrFileName))
             then (SETQ NoteFile (NC.NoteFileFromFileName FullFileName)))
         (OR (MEMBER NoteFile (NC.ListOfNoteFiles))
             (MEMBER FullFileName NCP.NoticedNoteFileNames])

)

;;; Stuff for dealing with the hash array.

(DEFINEQ

(NC.InstallCardInNoteFile
  [LAMBDA (Card NoteFile)                                             (* rht%: " 2-May-87 17:15")
```

(* * Put Card into NoteFile's hash array.)

(* * rht 5/2/87%: Now dies if there's already a card in the hash array having same UID.)

```
    (LET ((OtherCard (NC.CardFromUID (fetch (Card UID) of Card)
                                     NoteFile)))
         (if OtherCard
             then (SHOULDNT "Two cards in notefile with same UID.  Please notify the nearest NoteCards guru.")
           else (PUTHASH (fetch (Card UID) of Card)
                         Card
                         (fetch (NoteFile HashArray) of NoteFile))
                Card])
```

(NC.CardFromUID

```
  [LAMBDA (UID NoteFile)                                              (* pmi%: " 8-Sep-87 17:30")
```

(* * Recover the card with given UID by looking it up in the notefile's hash table.)

(* * pmi 9/8/87%: Now returns NIL if either UID or NoteFile are NIL.)

```
    (AND UID NoteFile (GETHASH UID (fetch (NoteFile HashArray) of NoteFile])

)

(DEFINEQ

(NC.MakeHashKey
  [LAMBDA (UID)                                                       (* fgh%: "20-Nov-85 19:25")
```

(* * make a hash key from a UID)

```
    (COND
       ((NOT (type? UID UID))
        (ERROR UID "Not a UID"))
```

```
       (T (LOGXOR (ffetch (UID UID0) of UID)
                  (ffetch (UID UID1) of UID)
                  (ffetch (UID UID2) of UID)
                  (ffetch (UID UID3) of UID)
                  (ffetch (UID UID4) of UID)
                  (ffetch (UID UID5) of UID)
                  (ffetch (UID UID6) of UID])
```

(NC.CreateUIDHashArray

```
  [LAMBDA (MinKeys OverFlow)                                    (* fgh%: "20-Nov-85 18:59")
    (HASHARRAY MinKeys OverFlow (FUNCTION NC.MakeHashKey)
           (FUNCTION NC.SameUIDP])

)

(RPAQ? NC.NoteFilesHashArray (NC.CreateUIDHashArray NC.NoteFilesHashArraySize))
```

;;; Stuff for dealing with CardLocs.

```
(DEFINEQ
```

(NC.SetStatus

```
  [LAMBDA (Card Status)                                         (* rht%: "14-Nov-85 18:25")

          (* * Set the status field of the given Card)

    (replace (Card IndexDirtyFlg) of Card with T)
    (replace (Card Status) of Card with Status])
```

(NC.SetMainLoc

```
  [LAMBDA (Card MainLoc)                                        (* rht%: "14-Nov-85 18:26")

          (* * Set the MainLoc field of the given cardID.)

    (replace (Card IndexDirtyFlg) of Card with T)
    (replace (Card MainLoc) of Card with MainLoc])
```

(NC.SetLinksLoc

```
  [LAMBDA (Card LinksLoc)                                       (* rht%: "14-Nov-85 18:26")

          (* * Set the LinksLoc field of the given cardID.)

    (replace (Card IndexDirtyFlg) of Card with T)
    (replace (Card LinksLoc) of Card with LinksLoc])
```

(NC.SetTitleLoc

```
  [LAMBDA (Card TitleLoc)                                       (* rht%: "14-Nov-85 18:27")

          (* * Set the TitleLoc field of the given Card)

    (replace (Card IndexDirtyFlg) of Card with T)
    (replace (Card TitleLoc) of Card with TitleLoc])
```

(NC.SetPropListLoc

```
  [LAMBDA (Card PropListLoc)                                    (* rht%: "14-Nov-85 18:27")

          (* * Set the PropListLoc field of the given Card)

    (replace (Card IndexDirtyFlg) of Card with T)
    (replace (Card PropListLoc) of Card with PropListLoc])

)
```

;;; The version object.

```
(RPAQ? NC.NoteFileVersionsList (LIST (create NoteFileVersion Version _ 3 NumberOfReservedCards _ 20
                                             NoteFileIndexWidth _ 28 NoteFileHeaderSize _ 30)))

(DEFINEQ
```

(NC.FetchCurrentVersionObject

```
  [LAMBDA NIL                                                   (* rht%: "15-Nov-85 01:16")

          (* * Return the NoteFileVersion object corresponding to the latest release of notecards.)

    (for NoteFileVersion in NC.NoteFileVersionsList when (EQ (fetch (NoteFileVersion Version) of NoteFileVersion)
                                                             NC.VersionNumber)
       do (RETURN NoteFileVersion])

)
```

;;; Stuff for copying cards from one notefile to another, or to the same.

```
(DEFINEQ
```

(NC.CopyCards

```
  [LAMBDA (Cards DestNoteFileOrFileBox RootCards QuietFlg InterestedWindow CopyExternalToLinksMode)
                                                              ; Edited  4-Aug-88 14:10 by Trigg
```

;;; Create copies of cards in Cards. If DestNoteFileOrFileBox is a notefile, then destination will be the contents box in that notefile, else the FileBox's
;;; notefile. RootCards should be NIL or a subset of Cards. If NIL, then file all Cards in the dest filebox. Otherwise, just file RootCards in that filebox and
;;; assume others are linked somehow to the RootCards. Links between cards in Cards are copied, but links from or to outside cards aren't.

;;; Currently all Cards must be in same notefile, but this perhaps could be relaxed if could prevent possibility of two cards in different notefiles having the
;;; same UID.

;; kirk 24Apr86 Added calls to select cards if none provided

;; rht 9/2/86: Added InterestedWindow arg.

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

;; rg 3/18/87 added NC.CardSelectionOperation wrapper

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

;; rht&rg&pmi 4/22/87: No longer calls ERROR!

;; rg 6/2/87 was checking for CANCELLED instead of DON'T

;; rg 6/5/87 deletes new cards if we cancel out halfway through

;; rht 6/6/87: Now optionally copies 'external' links. Passes extra new args to NC.FixUpLinksInCardCopy.

;; rht 6/22/87: Now returns list of cards copied, like it used to.

;; pmi 10/29/87: Now returns list of card copies, instead of cards copied.

;; pmi 12/10/87: Now returns new cards in the same order as their corresponding original cards. At dsj's suggestion (and implementation), now can
;; pass (QUOTE NONE) as RootCards, meaning don't file any of the new cards in the destination filebox.

;; dsj. 2/23/88. Fixed bug with (QUOTE NONE) arg.

;; rht 7/21/88: When asking for dest filebox, now passes NC.FileBoxP predicate to NC.SelectNoteCards. Also now calls card type's
;; ReplaceUIDsFn in case the card to copy has uid's appearing in its substance.

;; rht 7/29/88: No longer requires all cards to live in same notefile.

;; rht&pmi 8/4/88: Changed to print monitoring messages more often.

```
  (NCP.WithLockedCards (NC.IfAllCardsFree
                        (NC.LockListOfCards Cards "Copy Cards")
                        (PROG (NumCards SourceNoteFile DestNoteFile BoxToFileIn TempStream CardHashArray
                                    LinksHashArray CurrentLinkLabels NewLinkLabels NewCardsAndLocsOnStream
                                    CopyExternalToLinksFlg NewCardList)
                              ;; Make sure the arguments are valid.
                              (if (NULL Cards)
                                  then (if (NULL (SETQ Cards (NC.SelectNoteCards NIL NIL NC.SelectingCardsMenu
                                                                  NIL "Shift-select cards to copy:" NIL)))
                                           then (RETURN NIL)))
                              (SETQ Cards (MKLIST Cards))
                              (SETQ NumCards (LENGTH Cards))    ; All Cards to copy must live in open notefiles.
                              (for Card in (MKLIST Cards) bind SourceNoteFile
                                 unless (AND (type? NoteFile (SETQ SourceNoteFile (fetch (Card NoteFile)
                                                                                     of Card)))
                                             (OPENP (fetch (NoteFile Stream) of SourceNoteFile)))
                                 do (NC.ReportError "NC.CopyCards" (CONCAT (fetch (NoteFile FullFileName)
                                                                              of SourceNoteFile)
                                                                           " not an open notefile.")))
                                                                ; Compute dest notefile and dest filebox.
                              (if (NOT DestNoteFileOrFileBox)
                                  then (if (EQ 'DON'T (SETQ DestNoteFileOrFileBox (NC.SelectNoteCards
                                                                                    T
                                                                                    (FUNCTION NC.FileBoxP)
                                                                                    NC.SelectingCardMenu NIL
                                                                                    "Shift-select the FileBox to
                                                                                    contain these cards." T)))
                                           then (RETURN NIL)))
                              (if (type? NoteFile DestNoteFileOrFileBox)
                                  then (SETQ DestNoteFile DestNoteFileOrFileBox)
                                       (SETQ BoxToFileIn (fetch (NoteFile TableOfContentsCard) of DestNoteFile))
                                elseif (NCP.FileBoxP DestNoteFileOrFileBox)
                                  then (SETQ BoxToFileIn DestNoteFileOrFileBox)
                                       (SETQ DestNoteFile (fetch (Card NoteFile) of BoxToFileIn))
                                else (NC.ReportError "NC.CopyCards" (CONCAT "Arg not notefile or filebox: "
                                                                            DestNoteFileOrFileBox)))
                              (if [NOT (AND (type? NoteFile DestNoteFile)
                                            (OPENP (fetch (NoteFile Stream) of DestNoteFile]
                                  then (NC.ReportError "NC.CopyCards" (CONCAT (fetch (NoteFile FullFileName)
                                                                                 of DestNoteFile)
                                                                              " not an open notefile.")))
```

;;; dsj. Fixed bug here by switching the order of the next two Sexprs.

```
(if (NULL RootCards)
    then (SETQ RootCards Cards)
  elseif (EQ RootCards 'NONE)
    then (SETQ RootCards))
(if (LDIFFERENCE (SETQ RootCards (MKLIST RootCards))
          Cards)
    then (NC.ReportError "NC.CopyCards" "RootCards argument not subset of Cards
                argument."))
```

;;; Figure out whether to copy 'external' links.

```
[SETQ CopyExternalToLinksFlg (SELECTQ CopyExternalToLinksMode
                                   (COPY T)
                                   (DON'TCOPY NIL)
                                   (SELECTQ (NC.AskUserWithMenu
                                            '(Yes No Cancel)
                                            (CONCAT "You've asked to copy "
                                                   (LENGTH Cards)
                                                   " cards."
                                                   (CHARACTER 13)
                                                   "Links among these cards will
                                                   be automatically copied."
                                                   (CHARACTER 13)
                                                   "Do you also want to copy
                                                   links pointing from these
                                                   cards to elsewhere? ")
                                            InterestedWindow)
                                        (Yes T)
                                        (No NIL)
                                        (RETURN NIL]
```

;;; Now get to work.

```
(SETQ TempStream (OPENSTREAM '{NODIRCORE} 'BOTH))
(SETQ CurrentLinkLabels (NC.RetrieveLinkLabels DestNoteFile))
(SETQ NewLinkLabels (TCONC NIL))
(SETQ LinksHashArray (HASHARRAY NC.CopyCardsLinksHashArraySize NIL
                                (FUNCTION NC.MakeHashKey)
                                (FUNCTION NC.SameUIDP)))
(SETQ CardHashArray (HASHARRAY NumCards NIL (FUNCTION NC.MakeHashKeyFromCard)
                               (FUNCTION NC.SameCardP)))
```

;; Create new cards in DestNoteFile for each card. Make these cards by copying original cards to a temp stream.
;; Keep track of UID mappings between original cards and card copies using CardHashArray.

```
(OR QuietFlg (NC.PrintMsg InterestedWindow T "Copying cards: creating empty
                    copies." (CHARACTER 13)
                    "Processing item " 1 " out of " NumCards "..." (CHARACTER
                                                                    13)))
(SETQ NewCardsAndLocsOnStream NIL)
(RESETLST
    [RESETSAVE NIL '(PROGN (if RESETSTATE
                               then (for CardAndLoc in NewCardsAndLocsOnStream
                                       do (NC.DeleteNoteCardInternal
                                            (CAR CardAndLoc)
                                            T InterestedWindow))
                                    (NC.ClearMsg InterestedWindow T]
    (for Card in Cards as i from 1 bind NewCard WasActiveFlg HadStatusNILFlg
                                        IndexLocs
       eachtime (BLOCK) unless (NC.CrossFileLinkCardP Card)
       do [OR QuietFlg (if (ZEROP (REMAINDER i 10))
                           then (NC.PrintMsg InterestedWindow T "Copying cards:
                                    creating empty copies." (CHARACTER 13)
                                    "Processing item " i " out of " NumCards
                                    "..." (CHARACTER 13]
          (if (NOT (SETQ WasActiveFlg (NC.ActiveCardP Card)))
              then (NC.GetNoteCard Card))
          (if (SETQ HadStatusNILFlg (NULL (fetch (Card Status) of Card)))
              then                  ; Have to have Status slot ACTIVE in order that Put to stream
                                    ; won't break.
                   (replace (Card Status) of Card with 'ACTIVE))
          (SETQ IndexLocs (NC.PutNoteCardToStream Card NIL T TempStream))
          (if HadStatusNILFlg
              then (replace (Card Status) of Card with NIL))
          (if (NOT WasActiveFlg)
              then (NC.DeactivateCard Card))
                                    ; Make new empty card for copy.
          (SETQ NewCard (NC.GetNewCard DestNoteFile))
                                    ; Map old cards to card copies.
          (PUTHASH Card NewCard CardHashArray)
          (push NewCardsAndLocsOnStream (CONS NewCard IndexLocs))))
```

;; For each card, get it off the temp stream, fix its links, fix browser info if necessary, and write it down to the dest
;; notefile.

```
(SETFILEPTR TempStream 0)
(OR QuietFlg (NC.PrintMsg InterestedWindow T "Copying cards: fixing links and
                    UIDs." (CHARACTER 13)
                    "Processing item " 1 " out of " NumCards "..." (CHARACTER
```

```
                                                                                    13)))
                    (SETQ NewCardList)
                    (for NewCardAndLocsOnStream in NewCardsAndLocsOnStream as i from 1
                       eachtime (BLOCK) bind (CrossFileLinkModePropList _ (LIST DestNoteFile NIL))
                       do [OR QuietFlg (if (ZEROP (REMAINDER i 10))
                                           then (NC.PrintMsg InterestedWindow T "Copying cards:
                                                                fixing links and UIDs." (CHARACTER 13)
                                                                "Processing item " i " out of " NumCards
                                                                "..." (CHARACTER 13]
                          (LET ((NewCard (CAR NewCardAndLocsOnStream))
                                (IndexLocs (CDR NewCardAndLocsOnStream)))
                                                        ; Have to make status active for Get fns to work.
                               (NC.SetStatus NewCard 'ACTIVE)
                               (NC.GetNoteCardFromStream NewCard TempStream IndexLocs)
                               (NC.FixUpLinksInCardCopy NewCard CardHashArray LinksHashArray
                                      CurrentLinkLabels NewLinkLabels InterestedWindow
                                      CopyExternalToLinksFlg CrossFileLinkModePropList)
                               (if (NC.IsSubTypeOfP (NC.FetchType NewCard)
                                                    'Browser)
                                   then (NC.FixUpBrowserCardCopy NewCard CardHashArray))
                               ;; Fix up the uid's if this card type's substance contains uid's of other cards.  (Ideally, the above call to
                               ;; NC.FixUpBrowserCardCopy would be done with ReplaceUIDsFn, but until that becomes a real slot
                               ;; of the card type, we can't be sure that card types inheriting from Browser would have the
                               ;; ReplaceUIDsFn prop.)
                               (LET [(ReplaceUIDsFn (GETPROP (NC.FetchType NewCard)
                                                             'ReplaceUIDsFn]
                                    (if ReplaceUIDsFn
                                        then (APPLY* ReplaceUIDsFn NewCard CardHashArray)))
                               (NC.PutNoteCard NewCard)
                               (push NewCardList NewCard)))
                    ;; Have to go back and replace any internal cross-file links with normal links.
                    (NC.FixUpCrossFileLinksInCardCopies NewCardList)
                    ;; Link RootCards under filebox in DestNotefile.
                    (OR QuietFlg (NC.PrintMsg InterestedWindow T "Copying cards: filing "
                                              (LENGTH RootCards)
                                              " new cards in "
                                              (NC.FetchTitle BoxToFileIn)
                                              "..."
                                              (CHARACTER 13)))
                    (AND RootCards (NC.FileBoxCollectChildren NIL BoxToFileIn
                                     (for RootCard in RootCards eachtime (BLOCK)
                                        collect (GETHASH RootCard CardHashArray))
                                     T))
                    ;; Put out any new link labels to the dest notefile.
                    (AND (SETQ NewLinkLabels (CDAR NewLinkLabels))
                         (NC.StoreLinkLabels DestNoteFile (APPEND NewLinkLabels CurrentLinkLabels)))
                    (OR QuietFlg (NC.ClearMsg InterestedWindow T))
                    (RETURN NewCardList])
```

(NC.MoveCards

```
  [LAMBDA (Cards DestNoteFileOrFileBox RootCards QuietFlg InterestedWindow CopyExternalToLinksMode)
                                                        ; Edited 28-Jul-88 10:03 by pmi
```

;;; Move cards into a filebox by copying and deleting.

;;; rht&rg&pmi 4/22/87: Took out ERROR!

;;; rg 6/2/87 added NCP.WithLockedCards wrapper

;;; pmi 10/29/87: Added CopyExternalToLinksMode argument to be passed down to NC.CopyCards.

;;; pmi 7/28/88: Now returns new cards instead of old ones.

```
    (DECLARE (GLOBALVARS NC.SelectingCardsMenu))
    (LET (NewCards)
         (NCP.WithLockedCards (NC.IfAllCardsFree (NC.LockListOfCards Cards "Move Cards")
                                 (OR Cards (SETQ Cards (NC.SelectNoteCards NIL NIL NC.SelectingCardsMenu NIL
                                                          "Shift-select from the same NoteFile cards to
                                                          move:")))
                                 (if Cards
                                     then (SETQ Cards (MKLIST Cards))
                                          (SETQ NewCards (NC.CopyCards Cards DestNoteFileOrFileBox RootCards
                                                           QuietFlg InterestedWindow
                                                           CopyExternalToLinksMode))
                                          (NC.DeleteNoteCards Cards T NIL InterestedWindow QuietFlg NIL))
                                 NewCards])
```

(NC.PutNoteCardToStream

```
  [LAMBDA (Card UpdateUpdateListFlg UseOldDateFlg Stream)       (* rht%: "28-Jan-86 14:12")
```

(* * Put all the card parts of Card down to Stream and return an IndexLocs record containing locations of each of the card parts just written down.)

```
    (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
        (LET (MainCardDataLoc LinksLoc TitleLoc PropListLoc)
             (SETQ MainCardDataLoc (GETFILEPTR Stream))
             (NC.PutMainCardData Card UpdateUpdateListFlg UseOldDateFlg Stream)
             (SETQ LinksLoc (GETFILEPTR Stream))
             (NC.PutLinks Card UseOldDateFlg Stream)
             (SETQ TitleLoc (GETFILEPTR Stream))
             (NC.PutTitle Card UseOldDateFlg Stream)
             (SETQ PropListLoc (GETFILEPTR Stream))
             (NC.PutPropList Card UseOldDateFlg Stream)
             (create IndexLocs
                    MainCardDataLoc _ MainCardDataLoc
                    LinksLoc _ LinksLoc
                    TitleLoc _ TitleLoc
                    PropListLoc _ PropListLoc)))])
```

(NC.GetNoteCardFromStream

```
  [LAMBDA (Card Stream IndexLocs)                                    (* rht%: "28-Jan-86 14:17")
```

(* * Like NC.GetNoteCard except gets card from given Stream instead of its notefile.
Uses IndexLocs record to know where to look for card parts.)

```
    (WITH.MONITOR (NC.FetchMonitor (fetch (Card NoteFile) of Card))
        (SETFILEPTR Stream (fetch (IndexLocs MainCardDataLoc) of IndexLocs))
        (NC.GetMainCardData Card Stream)
        (SETFILEPTR Stream (fetch (IndexLocs LinksLoc) of IndexLocs))
        (NC.GetLinks Card Stream)
        (SETFILEPTR Stream (fetch (IndexLocs TitleLoc) of IndexLocs))
        (NC.GetTitle Card Stream)
        (SETFILEPTR Stream (fetch (IndexLocs PropListLoc) of IndexLocs))
        (NC.GetPropList Card Stream)
        (NC.ActivateCard Card)
       Card)])
```

(NC.MakeHashKeyFromCard

```
  [LAMBDA (Card)                                                     (* rht%: "28-Jan-86 15:54")
```

(* * Create a hash key from the card's UID.)

```
    (NC.MakeHashKey (fetch (Card UID) of Card])

)

(DEFINEQ
```

(NC.CopyCardPart

```
  [LAMBDA (FromStream ToStream FromPtr ToPtr InterestedWindow)    ; Edited  8-Dec-87 12:55 by Gobbel
    ;; Copy a card part from one stream to another
    ;; This function is used by both Compact to new file and CompactInPlace.  Returns Length The Length result is used by Compact to new file as a
    ;; success flag.  Length is used by CompactInPlace to compute the next place to write.
    ;; rht 11/3/86: Added flashw before error message.
    ;; rht 1/22/87: Removed unused CardPartTypeNum and Card args.  Added InterestedWindow arg.
    ;; rg 12/8/87: Call to obsolete fn ERRORN replaced by *
    (LET (Length EndPtr)
         (SETFILEPTR FromStream FromPtr)
         (SETFILEPTR ToStream ToPtr)
         (if (SETQ Length (FIXP (NC.ReadPtr FromStream 3)))
             then (SETQ EndPtr (PLUS FromPtr Length))
                  [until (if (CAR (ERSETQ (OR (COPYBYTES FromStream ToStream FromPtr EndPtr)
                                              0)))
                             then (RETURN Length)
                           else (if (STREQUAL (REPORT-CONDITION *LAST-CONDITION* InterestedWindow)
                                              "file system resources exceeded")
                                    then                                    ; file system resources exceeded
                                         (ERROR "Trouble copying card." (CONCAT "Try freeing at least "
                                                                                (MAX 1 (IQUOTIENT (IDIFFERENCE
                                                                                                    EndPtr
                                                                                                    FromPtr)
                                                                                                  512))
                                                                                " pages in "
                                                                                (FILENAMEFIELD ToStream 'HOST)
                                                                                (FILENAMEFIELD ToStream 'DIRECTORY)
                                                                                "."
                                                                                (CHARACTER 13)
                                                                                "Then click here, type OK, and hit
                                                                                RETURN"))
                                                NIL
                                  else (RETURN NIL]
           else (FLASHW InterestedWindow)
                (NC.PrintMsg InterestedWindow T "Bad NoteFile. Please Inspect and Repair." (CHARACTER 13))
                NIL])
```

(NC.ExpandIndexInPlace

```
[LAMBDA (NoteFile NewIndexSize TempStream InterestedWindow OperationMsg QuietFlg)
                                                                (* rht%: "24-May-87 00:36")
```

(* * Make room for a bigger index by copying a few card parts out to the end of the file.
Assumes a checkpoint has been done to write all information onto the file.)

(* * kirk |9/22/86| Changed to use NCLocalDevice fns)

(* * rht 11/3/86%: Added InterestedWindow and OperationMsg arg and fixed typos.)

(* * rht 1/22/87%: Was ignoring its TempStream argument. Now calls NC.CopySortedCardPartInPlaceToEOF instead of NC.CopySortedCardPartInPlace and now checks that it succeeded before continuing.)

(* * rht 5/15/87%: Completely rewrote to no longer sort card parts.
Now searches in file for next card part to move ala Inspect&repair.)

(* * rht 5/24/87%: Added QuietFlg arg.)

```
  (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow NoteFile)))
  [OR OperationMsg (SETQ OperationMsg (CONCAT "Expanding Index" (CHARACTER 13]
  [OR TempStream (SETQ TempStream (OPENSTREAM '{NODIRCORE} 'BOTH NIL '((TYPE BINARY]
  (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
        (OldTotalIndexSize (NC.TotalIndexSize (fetch (NoteFile HashArraySize) of NoteFile)))
        (NewTotalIndexSize (NC.TotalIndexSize NewIndexSize))
        CardPartFileLocs FileLoc TotalNewIndexEntries)
       (OR QuietFlg (NC.PrintMsg InterestedWindow T OperationMsg "Making room for expanded index."
                           (CHARACTER 13)
                           "Moving card part " 1 "." (CHARACTER 13)))
       (if (GREATERP NewTotalIndexSize (GETEOFPTR Stream))
           then
```

(* Need to lengthen the file so that copies of cards will be out of way of expanding index.)

```
               (SETFILEPTR Stream NewTotalIndexSize))
```

(* * Search for a card part and copy it out to end of file. Repeat till we've freed up enough room for the desired number of new index entries. FileLoc winds up pointing to new start of data area.)

```
       (SETQ FileLoc OldTotalIndexSize)
       [SETQ CardPartFileLocs (for CTR from 1 eachtime (BLOCK) bind CardPartRecord ToPtr
                                 while [AND (SETQ CardPartRecord (NC.AutoloadApply* (FUNCTION
                                                                                    NC.FindNextCardPart)
                                                                                NoteFile FileLoc))
                                            (LESSP (SETQ FileLoc (fetch (CardPartRecord FileLoc) of
                                                                                                CardPartRecord
                                                                                               ))
                                                   NewTotalIndexSize)
                                            (PROGN [OR QuietFlg (if (ZEROP (IREMAINDER CTR 100))
                                                                    then (NC.PrintMsg InterestedWindow T
                                                                              OperationMsg "Making room for
                                                                              expanded index." (CHARACTER
                                                                                                13)
                                                                              "Moving card part " CTR "."
                                                                              (CHARACTER 13]
                                                   (SETQ ToPtr (NC.CopyCardPartInPlaceToEOF NoteFile
                                                                  CardPartRecord TempStream
                                                                  InterestedWindow]
                                 collect
```

(* Put out the new ChkptPtr to the file just in case we crash inside this loop.)

```
                                      (NC.PutCheckptPtr NoteFile ToPtr)
                                      (PROG1 FileLoc
                                          (SETQ FileLoc (PLUS FileLoc (fetch (CardPartRecord CardPartLength)
                                                                         of CardPartRecord))))]
```

(* * Compute the number of new entries we now have space to accomodate.
May be less than was asked for if we bombed in middle of copy.)

```
       [SETQ TotalNewIndexEntries (QUOTIENT (DIFFERENCE FileLoc OldTotalIndexSize)
                                            (CONSTANT (fetch (NoteFileVersion NoteFileIndexWidth) of (
                                                                                  NC.FetchCurrentVersionObject
                                                                                                    ]
       (PROG1 (if (GEQ TotalNewIndexEntries 1)
                  then
```

(* * We at least got room for one new index entry, so record new index size in file and write down hash array.)

```
                      (SETQ NewIndexSize (PLUS (fetch (NoteFile HashArraySize) of NoteFile)
                                               TotalNewIndexEntries))
                      (NCLocalDevice.PutHashArray NoteFile InterestedWindow NIL OperationMsg QuietFlg)
                      (replace (NoteFile HashArraySize) of NoteFile with NewIndexSize)
                                                                (* Make sure new hash array size gets written down.)
                      (NC.PutNoteFileHeader NoteFile)
```

```
(* * An ugly kludge%: must smash old %### indicators in file for newly copied card parts with 0's so no one will accidentally
back up to them using inspector. Those old card parts are now in index territory.
Had to wait until PutHashArray succeeded before doing this.)

                    (for OldFileLoc in CardPartFileLocs when (AND (NUMBERP OldFileLoc)
                                                                  (LESSP OldFileLoc (GETEOFPTR Stream)))
                       do (SETFILEPTR Stream OldFileLoc)
                          (NC.WritePtr Stream 0 6))          (* Move index from old hash array into larger hash array.)
                    (LET ((NewHashArray (NC.CreateUIDHashArray NewIndexSize)))
                         (REHASH (fetch (NoteFile HashArray) of NoteFile)
                                 NewHashArray)
                         (replace (NoteFile HashArray) of NoteFile with NewHashArray))
                    TotalNewIndexEntries
              else

(* * We weren't able to recover room for any new index entries.)

                 NIL)
       (OR QuietFlg (NC.ClearMsg InterestedWindow T)))])
```

(NC.CheckForExpandIndex

```
  [LAMBDA (NoteFile QuietFlg InterestedWindow)                    (* Randy.Gobbel " 5-Jun-87 12:21")

(* * If index is full, then confirm with user whether it's okay to expand in place.
Offer user ability to change the amount to expand by. Return nil if no expand, return new index size otherwise.
We MUST expand otherwise error out.)

(* * rht 5/24/87%: Now passes QuietFlg to NC.ExpandIndexInPlace.)

(* * rg |6/3/87| call to Checkpoint wasn't checking for DON'T)

(* * rg |6/5/87| now only offers to expand if completely full, plus misc other small fixes)

    (DECLARE (GLOBALVARS NC.IndexFractionToExpandBy NC.MenuFont))
    (OR InterestedWindow (SETQ InterestedWindow (NC.CoerceToInterestedWindow NoteFile)))
    (LET [(IndexSize (fetch (NoteFile HashArraySize) of NoteFile))
          (NumUsed (SUB1 (DIFFERENCE (fetch (NoteFile NextIndexNum) of NoteFile)
                                     (LENGTH (fetch (NoteFile IndexNumsFreeList) of NoteFile]
         (if (EQ NumUsed IndexSize)
             then (LET ([Menu (create MENU
                                      TITLE _ " Expand notefile index? "
                                      CENTERFLG _ T
                                      MENUBORDERSIZE _ 1
                                      MENUOUTLINESIZE _ 1
                                      MENUROWS _ 1
                                      MENUFONT _ NC.MenuFont
                                      ITEMHEIGHT _ (IPLUS 10 (FONTPROP NC.MenuFont 'HEIGHT))
                                      MENUTITLEFONT _ (FONTCREATE 'HELVETICA 10 'BOLD)
                                      ITEMS _ '((Yes 'Yes "Go ahead and expand index to make room for new cards.")
                                                (Cancel 'Cancel "Cancel creation of new card.")
                                                (Change% Num 'Change% Num "Change the number of new cards to
                                                      expand index by."]
                              (PromptWindow (OR (NC.AttachPromptWindow InterestedWindow)
                                                PROMPTWINDOW))
                              NewIndexSize NumNewSlots NewNum ActualNumNewSlots)
                             [SETQ NewIndexSize (PLUS IndexSize (SETQ NumNewSlots (FIX (FTIMES IndexSize
                                                                                               NC.IndexFractionToExpandBy
                                                                                               ]
                             (for do (NC.PrintMsg InterestedWindow T (CONCAT (fetch (NoteFile FullFileName) of NoteFile)
                                                                             " is full (" NumUsed " out of " IndexSize
                                                                             " cards used)." (CHARACTER 13)))
                                   (NC.PrintMsg InterestedWindow NIL (CONCAT "Okay to checkpoint notefile and make
                                                                             room for " NumNewSlots " new cards?"
                                                                             (CHARACTER 13)))
                                   (ALLOW.BUTTON.EVENTS)
                                   (SELECTQ (MENU Menu (CREATEPOSITION (fetch (REGION LEFT) of (WINDOWREGION
                                                                                                PromptWindow))
                                                                       (fetch (REGION TOP) of (WINDOWREGION PromptWindow)))
                                                       T)
                                       (Yes                                  (* Go ahead and expand index after checkpointing notefile.)
                                            (if (EQ (NC.CheckpointNoteFile NoteFile NIL NIL InterestedWindow
                                                                           (CONCAT "Expanding notefile index" (CHARACTER 13)))
                                                    'DON'T)
                                                then (NC.ClearMsg InterestedWindow T)
                                                     (ERROR!))
                                            (SETQ ActualNumNewSlots (NC.ExpandIndexInPlace NoteFile NewIndexSize NIL
                                                                                           InterestedWindow NIL QuietFlg))
                                            (COND
                                               ((NULL ActualNumNewSlots)
                                                                         (* Looks like NC.ExpandIndexInPlace bombed for some reason.)
                                                (FLASHW PROMPTWINDOW) (* If index is full, we have to bail out of card creation.)
                                                (NC.PrintMsg NIL T "Couldn't expand index.  Card creation operation
                                                       canceled.")
                                                (ERROR!))
                                               ((LESSP ActualNumNewSlots NumNewSlots)
```

```
                                                    (* NC.ExpandIndexInPlace probably bombed but we at least got
                                                    some of what we asked for.)
                                                    (FLASHW PROMPTWINDOW)
                                                    (NC.PrintMsg NIL T "Only able to make room for " ActualNumNewSlots
                                                           " new cards out of " NumNewSlots " requested." NIL)))
                                               (NC.ClearMsg InterestedWindow T)
                                               (RETURN NewIndexSize))
                                          (Cancel                    (* Bad news. We have to bail out because index is full.)
                                                  (NC.ClearMsg InterestedWindow T)
                                                  (ERROR!))
                                          (Change% Num               (* Let user change number of new slots to make room for.)
                                                       (SETQ NewNum (RNUMBER "Number of new cards to add" NIL NIL NIL T)
                                                        )
                                                       [if (AND (NUMBERP NewNum)
                                                                (GREATERP NewNum 0))
                                                           then (SETQ NewIndexSize (PLUS IndexSize (SETQ NumNewSlots
                                                                                                     NewNum])
                                          NIL])
```

(NC.FindNextCardPart

```
[LAMBDA (NoteFile StartPtr)                                    (* rht%: "16-May-87 00:59")

        (* * Start searching at the current stream pointer in NoteFile for the next card part.
        Return a "CardPartRecord" if successful containing file loc, card uid and card part type number.
        This code largely stolen from NC.RobustReadCardPart from NCREPAIR.
        Notice the calls to NCREPAIR fns.)

   (DECLARE (GLOBALVARS NC.TitlesIdentifier NC.PropsIdentifier NC.LinksIdentifier NC.ItemIdentifier))
   (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
         EndPtr CardPartPtr IdentifierAndVersionNum IdentifierAtom UID CardPartLength CardPartTypeNum)
        (SETQ EndPtr (GETEOFPTR Stream))
        (SETQ CardPartPtr (NC.SearchFor### Stream StartPtr))
        (for
           do                                                  (* Keep looping till we find a reasonable card part.)
               (if (AND (SETQ CardPartLength (NC.ReadPtr Stream 3))
                        (LEQ (PLUS CardPartPtr CardPartLength)
                             EndPtr)
                        (SETQ IdentifierAndVersionNum (NC.RobustReadItemIdentifier Stream))
                        (if (GEQ (CDR IdentifierAndVersionNum)
                                 1)
                            then (NC.RobustReadDate Stream)
                          else T)
                        (SETQ UID (NC.RobustReadUID Stream)))
                   then (SETQ IdentifierAtom (CAR IdentifierAndVersionNum))
                        (SETQ CardPartTypeNum (COND
                                                 ((EQ IdentifierAtom NC.ItemIdentifier)
                                                  0)
                                                 ((EQ IdentifierAtom NC.LinksIdentifier)
                                                  1)
                                                 ((EQ IdentifierAtom NC.TitlesIdentifier)
                                                  2)
                                                 ((EQ IdentifierAtom NC.PropsIdentifier)
                                                  3)))
                        (RETURN (create CardPartRecord
                                        FileLoc _ CardPartPtr
                                        UID _ UID
                                        CardPartTypeNum _ CardPartTypeNum
                                        CardPartLength _ CardPartLength))
                 else (if [NULL (SETQ CardPartPtr (NC.SearchFor### Stream (PLUS CardPartPtr 4]
                          then                                 (* We've reached EOF without finding a card part.)
                               (RETURN NIL])
```

(NC.SearchFor###

```
[LAMBDA (Stream Ptr)                                           (* rht%: " 2-Dec-85 16:28")

        (* * Move the file ptr to next occurrence of either %# or NOBIND.
        The latter is for the stupid case of NOBIND instead of titles identifier.
        The choice of FFILEPOS rather than FILEPOS for the NOBIND search is based on empirical evidence from TIMEALL.)

        (* * rht 12/1/85%: Now positions file 3 bytes in front of %###, to account for new length bytes.
        Now doesn't fool with NOBIND litatoms.)

   (LET ((NewPtr (FILEPOS '%### Stream Ptr)))
        (AND NewPtr (SETFILEPTR Stream (DIFFERENCE NewPtr 3])
```

(NC.RobustReadItemIdentifier

```
[LAMBDA (Stream)                                               (* rht%: " 1-Dec-85 22:24")

        (* * Look for an item identifier at the current position in Stream.
        If successful, return the part/item type.)

        (* * rht 12/1/85%: Ripped out kludgy NOBIND litatom test.)

   (LET ((OldPtr (GETFILEPTR Stream))
```

```
         Atom VersionNumber)
        (SETQ Atom (CAR (NC.RobustRead Stream)))
        (if [AND (FMEMB Atom NC.IdentifierAtoms)
                 (NC.RobustReadChar Stream)
                 (NUMBERP (SETQ VersionNumber (CAR (NC.RobustReadByte Stream]
            then (CONS Atom VersionNumber)
          else (SETFILEPTR Stream OldPtr)
               NIL])
```

(NC.RobustReadDate

```
[LAMBDA (Stream)                                                    (* rht%: " 1-Dec-85 22:32")
```

(* * Try to read a date string or the litatom NIL. Return a list containing the date or NIL indicating failure.)

(* * rht 12/1/85%: Now skips past CR if successful.)

```
  (LET ((OldPtr (GETFILEPTR Stream))
        Val)
       (if (OR [NULL (SETQ Val (CAR (NC.RobustRead Stream]
               (AND (STRINGP Val)
                    (EQ (NCHARS Val)
                        NC.DateStringLength)))
           then                                                     (* Skip CR.)
                (NC.RobustReadChar Stream)
                (LIST Val)
         else (SETFILEPTR Stream OldPtr)
              NIL])
```

(NC.RobustReadUID

```
[LAMBDA (Stream)                                                    (* rht%: " 1-Dec-85 23:03")
```

(* * Try to read a Notecards ID from Stream. Return NIL if it's not a valid ID.)

(* * rht 12/1/85%: Updated to handle new card format.)

```
  (LET [(OldPtr (GETFILEPTR Stream))
        (Val (CAR (RESETVAR HELPFLAG NIL (NLSETQ (NC.ReadUID Stream]
       (if (type? UID Val)
           then Val
         else (SETFILEPTR Stream OldPtr)
              NIL])
```

(NC.RobustReadChar

```
[LAMBDA (Stream)                                                    (* rht%: "22-Mar-86 16:01")
```

(* * Try to read a character, In an NLSETQ so we won't see error messages.
The RESETVAR is so that no breaks will occur. This returns list of the one element read or NIL if unsuccessful read.)

```
  (RESETVAR HELPFLAG NIL (NLSETQ (CHARACTER (BIN Stream])
```

(NC.RobustReadByte

```
[LAMBDA (Stream)                                                    (* rht%: "10-Jul-85 23:07")
```

(* * Try to read a byte, In an NLSETQ so we won't see error messages.
The RESETVAR is so that no breaks will occur. This returns list of the one element read or NIL if unsuccessful read.)

```
  (RESETVAR HELPFLAG NIL (NLSETQ (NC.GetPtr Stream 1])
```

(NC.RobustRead

```
[LAMBDA (Stream)                                                    (* rht%: " 1-Nov-86 15:21")
```

(* * Try to read an object, In an NLSETQ so we won't see error messages.
The RESETVAR is so that no breaks will occur. This returns list of the one element read or NIL if unsuccessful read.)

(* * rht 11/1/86%: Now uses our readtable when reading.)

```
  (DECLARE (GLOBALVARS NC.OrigReadTable))
  (RESETVAR HELPFLAG NIL (NLSETQ (READ Stream NC.OrigReadTable])
```

(NC.CopyCardPartInPlaceToEOF

```
[LAMBDA (NoteFile CardPartRecord TempStream InterestedWindow)       (* rht%: "15-May-87 22:01")
```

(* * This copies given card part to the end of the notefile.)

```
  (LET ((FromPtr (fetch (CardPartRecord FileLoc) of CardPartRecord))
        (UID (fetch (CardPartRecord UID) of CardPartRecord))
        (CardPartTypeNum (fetch (CardPartRecord CardPartTypeNum) of CardPartRecord)))
       (WITH.MONITOR (NC.FetchMonitor NoteFile)
           (LET ((Stream (fetch (NoteFile Stream) of NoteFile))
                 EOFPtr Length)
                (SETQ EOFPtr (GETEOFPTR Stream))
```

```
        (* * Copy the substance out to the {NODIRCORE} stream.)

                (SETQ Length (NC.CopyCardPart Stream TempStream FromPtr 0 InterestedWindow))

        (* * Now copy to its proper home.)

                (if (NC.CopyCardPart TempStream Stream 0 EOFPtr InterestedWindow)
                    then (NC.UpdateIndexLocIfNeeded NoteFile UID CardPartTypeNum FromPtr EOFPtr)
                         (PLUS EOFPtr Length)))])
```

(NC.UpdateIndexLocIfNeeded

```
  [LAMBDA (NoteFile UID CardPartTypeNum OldLoc NewLoc)          (* rht%: "15-May-87 22:00")

        (* * See if the current index pointer for given UID's card part points to OldLoc.
        If so, then change to point to NewLoc. Else do nothing, the card part version is not currently in use.)

    (LET ((Card (NC.CardFromUID UID NoteFile)))
         (if Card
             then (AND (SELECTQ CardPartTypeNum
                                (0 (if (EQUAL (fetch (Card MainLoc) of Card)
                                              OldLoc)
                                       then (replace (Card MainLoc) of Card with NewLoc)))
                                (1 (if (EQUAL (fetch (Card LinksLoc) of Card)
                                              OldLoc)
                                       then (replace (Card LinksLoc) of Card with NewLoc)))
                                (2 (if (EQUAL (fetch (Card TitleLoc) of Card)
                                              OldLoc)
                                       then (replace (Card TitleLoc) of Card with NewLoc)))
                                (3 (if (EQUAL (fetch (Card PropListLoc) of Card)
                                              OldLoc)
                                       then (replace (Card PropListLoc) of Card with NewLoc)))
                                (PROGN (FLASHW PROMPTWINDOW)
                                       (NC.PrintMsg NIL T "Bad NoteFile. Please Inspect and Repair." (CHARACTER 13))
                                       NIL))
                       (replace (Card IndexDirtyFlg) of Card with T])

)

(DEFINEQ
```

(NC.FixUpLinksInCardCopy

```
  [LAMBDA (CardCopy CardHashArray LinksHashArray CurrentLinkLabels NewLinkLabels InterestedWindow
                CopyExternalToLinksFlg CrossFileLinkModePropList)
                                                          ; Edited 4-Aug-88 21:35 by Trigg

;;; For all the links from or to CardCopy, change other endpoint's card according to mapping table in CardHashArray. If other endpoint is a card not
;;; found in the hash array, then drop that link altogether. The mapping from old link UIDs to new ones is in LinksHashArray. Any new link labels not in
;;; CurrentLinkLabels get TCONC'ed onto NewLinkLabels.

    ;; rht 2/17/86: Now uses NC.ApplyFn instead of APPLY* for deleting and collecting references.

    ;; rht 11/1/86: Added missing var bindings and a BLOCK

    ;; rht 6/6/87: Added new args InterestedWindow, CopyExternalToLinksFlg, and CrossFileLinkModePropList to handle optional copying of external
    ;; links.

    ;; rht 7/29/88: Now handles case when dest card is a cross-file link card and remote dest card is one we're supposed to copy. That is, we replace
    ;; the cross-file link with a normal link between the copies.

    ;; rht 8/4/88: Now doesn't follow cross file links in order to check whether dest card is in set of copied cards.

    (LET ((CardCopyType (NC.FetchType CardCopy)))

        ;; Fix all the From links.

        (NC.SetFromLinks CardCopy (for Link in (NC.FetchFromLinks CardCopy) eachtime (BLOCK)
                                     bind SourceCard OldLinkUID LinkLabel
                                     when (SETQ SourceCard (GETHASH (fetch (Link SourceCard) of Link)
                                                                    CardHashArray))
                                     collect (replace (Link DestinationCard) of Link with CardCopy)
                                             (replace (Link SourceCard) of Link with SourceCard)
                                             (replace (Link UID) of Link with (OR (GETHASH (SETQ OldLinkUID
                                                                                                 (fetch (Link UID)
                                                                                                    of Link))
                                                                                           LinksHashArray)
                                                                                  (PUTHASH OldLinkUID (NC.MakeUID)
                                                                                           LinksHashArray)))
                                                     ; Keep track of link labels in case any are new.
                                             (OR (FMEMB (SETQ LinkLabel (fetch (Link Label) of Link))
                                                        CurrentLinkLabels)
                                                 (NC.SystemLinkLabelP LinkLabel)
                                                 (FMEMB LinkLabel (CAR NewLinkLabels))
                                                 (TCONC NewLinkLabels LinkLabel))
                                             Link))

        ;; Do it all again for the To links.

        (NC.SetToLinks CardCopy (for Link in (NC.FetchToLinks CardCopy) eachtime (BLOCK) bind DestCard OldLinkUID
                                                                                              LinkLabel
                                   when (SETQ DestCard (GETHASH (fetch (Link DestinationCard) of Link)
```

```
                                                    CardHashArray))
                                  collect (replace (Link SourceCard) of Link with CardCopy)
                                          (replace (Link DestinationCard) of Link with DestCard)
                                          (replace (Link UID) of Link with (OR (GETHASH (SETQ OldLinkUID
                                                                                              (fetch (Link UID)
                                                                                                 of Link))
                                                                                       LinksHashArray)
                                                                               (PUTHASH OldLinkUID (NC.MakeUID)
                                                                                       LinksHashArray)))
                                                                            ; Keep track of link labels in case any are new.
                                          (OR (FMEMB (SETQ LinkLabel (fetch (Link Label) of Link))
                                                     CurrentLinkLabels)
                                              (NC.SystemLinkLabelP LinkLabel)
                                              (FMEMB LinkLabel (CAR NewLinkLabels))
                                              (TCONC NewLinkLabels LinkLabel))
                                          Link))
```

;; Yet again for global links. Don't have to mess with link labels here 'cause ToLinks loop took care of that.

```
(NC.SetGlobalLinks CardCopy (for Link in (NC.FetchGlobalLinks CardCopy) eachtime (BLOCK) bind DestCard
                                                                                             OldLinkUID
                                  when (SETQ DestCard (GETHASH (fetch (Link DestinationCard) of Link)
                                                               CardHashArray))
                                  collect (replace (Link SourceCard) of Link with CardCopy)
                                          (replace (Link DestinationCard) of Link with DestCard)
                                          (replace (Link UID) of Link
                                             with (OR (GETHASH (SETQ OldLinkUID (fetch (Link UID) of Link))
                                                               LinksHashArray)
                                                      (PUTHASH OldLinkUID (NC.MakeUID)
                                                               LinksHashArray)))
                                          Link))
```

;; Now fix the links inside imageobj's in the card's substance.

```
(AND (fetch (Card LinkAnchorModesSupported) of CardCopy)
     (for Link in (CAR (NC.ApplyFn CollectLinksFn CardCopy)) eachtime (BLOCK) bind PreviousLink
        do (LET ((DestCard (fetch (Link DestinationCard) of Link))
                 (LinkLabel (fetch (Link Label) of Link))
                 OldLinkUID DestCardCopy NewLink)
                (COND
                   ((AND (NC.CrossFileLinkCardP DestCard)
                         (SETQ DestCardCopy (GETHASH (create Card
                                                             UID _ (fetch (CrossFileLinkSubstance
                                                                             CrossFileLinkDestCardUID
                                                                             )
                                                                     of (NCP.CardSubstance DestCard)))
                                                     CardHashArray)))
                                                    ; It's an internal cross-file link. Mark the card as needing its
                                                    ; cross-file link replaced by a non-cross-file link.
                    [NC.SetUserDataProp CardCopy 'CrossFileLinksToFix (CONS (LIST Link DestCardCopy)
                                                                            (NC.FetchUserDataProp
                                                                             CardCopy
                                                                             'CrossFileLinksToFix]
                                                    ; Keep track of link labels in case any are new.
                    (OR (FMEMB LinkLabel CurrentLinkLabels)
                        (NC.SystemLinkLabelP LinkLabel)
                        (FMEMB LinkLabel (CAR NewLinkLabels))
                        (TCONC NewLinkLabels LinkLabel))
                    (SETQ PreviousLink Link))
                   ((SETQ DestCardCopy (GETHASH DestCard CardHashArray))
                    (replace (Link SourceCard) of Link with CardCopy)
                    (replace (Link DestinationCard) of Link with DestCardCopy)
                    (replace (Link UID) of Link with (OR (GETHASH (SETQ OldLinkUID (fetch (Link UID)
                                                                                      of Link))
                                                                  LinksHashArray)
                                                         (PUTHASH OldLinkUID (NC.MakeUID)
                                                                  LinksHashArray)))
                    (SETQ PreviousLink Link))
                   ([AND CopyExternalToLinksFlg (if (NC.CrossFileLinkCardP DestCard)
                                                    then (SETQ DestCard (NC.GetCrossFileLinkDestCard
                                                                           DestCard InterestedWindow))
                                                  else DestCard)
                         (SETQ NewLink (NC.MakeLink NIL LinkLabel DestCard CardCopy (fetch (Link
                                                                                             DisplayMode
                                                                                             )
                                                                                    of Link)
                                                    (fetch (Link AnchorMode) of Link)
                                                    NIL NIL PreviousLink (NC.ComputeCrossFileLinkMode
                                                                            DestCard
                                                                            CrossFileLinkModePropList
                                                                            InterestedWindow]
                    ;; It's an external link. Try to make a copy, possibly resulting in a cross-file link.
                                                    ; Smash the imageobj's link with contents of new one we just
                                                    ; made.
                    [for FieldName in (RECORDFIELDNAMES 'Link)
                       do (RECORDACCESS FieldName Link (RECLOOK 'Link)
                                        'REPLACE
                                        (RECORDACCESS FieldName NewLink (RECLOOK 'Link)
                                                      'FETCH]       ; Keep track of link labels in case any are new.
```

```
                            (OR (FMEMB LinkLabel CurrentLinkLabels)
                                (NC.SystemLinkLabelP LinkLabel)
                                (FMEMB LinkLabel (CAR NewLinkLabels))
                                (TCONC NewLinkLabels LinkLabel))
                            (SETQ PreviousLink Link))
                           (T (NC.ApplyFn DeleteLinksFn CardCopy Link])
```

(NC.FixUpCrossFileLinksInCardCopies

```
[LAMBDA (Cards)                                          ; Edited 29-Jul-88 19:00 by Trigg
```

;; For each card in NewCards, see if it was marked as having internal cross-file links. If so, replace them with normal links.

```
(for Card in Cards bind CrossFileLinksToFix when (SETQ CrossFileLinksToFix (NC.FetchUserDataProp Card
                                                                            'CrossFileLinksToFix))
   do (for Link in (CAR (NC.ApplyFn CollectLinksFn Card)) eachtime (BLOCK) bind PreviousLink
         do (LET (NewDestCard NewLink)
                 [if [SETQ NewDestCard (for CrossFileLinkInfo in CrossFileLinksToFix
                                          when (NC.SameLinkP Link (CAR CrossFileLinkInfo))
                                          do (RETURN (CADR CrossFileLinkInfo]
                     then (SETQ NewLink (NC.MakeLink NIL (fetch (Link Label) of Link)
                                                NewDestCard Card (fetch (Link DisplayMode) of Link)
                                                (fetch (Link AnchorMode) of Link)
                                                NIL NIL PreviousLink))
                                                          ; Smash the imageobj's link with contents of new one we just
                                                          ; made.
                          (for FieldName in (RECORDFIELDNAMES 'Link)
                             do (RECORDACCESS FieldName Link (RECLOOK 'Link)
                                       'REPLACE
                                       (RECORDACCESS FieldName NewLink (RECLOOK 'Link)
                                              'FETCH]
                 (SETQ PreviousLink Link)))
      (NC.SetUserDataProp Card 'CrossFileLinksToFix NIL])
```

(NC.FixUpBrowserCardCopy

```
[LAMBDA (BrowserCard CardsHashArray)                     (* rht%: "20-Feb-86 12:23")
```

(* * Fix up the parts of the new browser card copy. Need to fix roots and graphnodes.)

```
(LET [(Graph (NC.FetchSubstance BrowserCard))
      (GraphNodeIDHashArray (HASHARRAY NC.CopyBrowserHashArraySize NIL (FUNCTION NC.MakeHashKey)
                                  (FUNCTION NC.SameUIDP]
```

(* * Fix up browser roots.)

```
(NC.SetBrowserRoots BrowserCard (for Card in (NC.FetchBrowserRoots BrowserCard)
                                   collect (GETHASH Card CardsHashArray)))
```

(* * Fix up graph nodes.)

```
[for GraphNode in (fetch (GRAPH GRAPHNODES) of Graph) eachtime (BLOCK)
   do (replace (GRAPHNODE NODEID) of GraphNode with (NC.BrowserCopyConvertGraphNodeID (fetch (GRAPHNODE
                                                                                              NODEID)
                                                                                       of GraphNode)
                                                                          GraphNodeIDHashArray))
      [replace (GRAPHNODE TONODES) of GraphNode
         with (for NodeID in (fetch (GRAPHNODE TONODES) of GraphNode)
                 collect (if (EQ (CAR NodeID)
                                 LINKPARAMS)
                             then (RPLACA (CDR NodeID)
                                          (NC.BrowserCopyConvertGraphNodeID (CADR NodeID)
                                                 GraphNodeIDHashArray))
                                  (AND (LISTGET NodeID 'NODEID)
                                       (LISTPUT NodeID 'NODEID (NC.BrowserCopyConvertGraphNodeID
                                                                  (LISTGET NodeID 'NODEID)
                                                                 GraphNodeIDHashArray)))
                                  (AND (LISTGET NodeID 'DESTNODEID)
                                       (LISTPUT NodeID 'DESTNODEID (NC.BrowserCopyConvertGraphNodeID
                                                                     (LISTGET NodeID 'DESTNODEID)
                                                                    GraphNodeIDHashArray)))
                                  NodeID
                             else (NC.BrowserCopyConvertGraphNodeID NodeID GraphNodeIDHashArray]
      (replace (GRAPHNODE FROMNODES) of GraphNode
         with (for NodeID in (fetch (GRAPHNODE FROMNODES) of GraphNode)
                 collect (if (EQ (CAR NodeID)
                                 LINKPARAMS)
                             then (RPLACA (CDR NodeID)
                                          (NC.BrowserCopyConvertGraphNodeID (CADR NodeID)
                                                 GraphNodeIDHashArray))
                                  NodeID
                             else (NC.BrowserCopyConvertGraphNodeID NodeID GraphNodeIDHashArray]
```

(* * Fix up the saved linking info.)

```
(for SavedLinkingInfoForNode in (NC.FetchBrowserSavedLinkingInfo BrowserCard)
   do (RPLACA SavedLinkingInfoForNode (NC.BrowserCopyConvertGraphNodeID (CAR SavedLinkingInfoForNode)
                                             GraphNodeIDHashArray))
```

```
            (for SavedLinkingInfo on (CDR SavedLinkingInfoForNode) by (CDDR SavedLinkingInfo)
               do (RPLACA SavedLinkingInfo (NC.BrowserCopyConvertGraphNodeID (CAR SavedLinkingInfo)
                                                   GraphNodeIDHashArray])
```

(NC.BrowserCopyConvertGraphNodeID

```
  [LAMBDA (NodeID GraphNodeIDHashArray)                    ; Edited  4-Aug-88 14:07 by Trigg
```

;;; Convert a graph node ID using the given hash array.

;; rht&pmi 8/4/88: Now makes sure it's a node for a card (e.g. not a label node).

```
    (if (NC.CardFromBrowserNodeID NodeID)
        then (if (LISTP NodeID)
                 then (LIST (OR (GETHASH (CAR NodeID)
                                         GraphNodeIDHashArray)
                                (PUTHASH (CAR NodeID)
                                         (NC.MakeBrowserNodeUID)
                                         GraphNodeIDHashArray)))
                 else (OR (GETHASH NodeID GraphNodeIDHashArray)
                          (PUTHASH NodeID (NC.MakeBrowserNodeUID)
                                   GraphNodeIDHashArray)))
      else NodeID])

)
```

;;; Traversal specs, should be in an NCTRAVERSAL module.

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.TraversalSpecsStylesheet)
)

(RPAQ? NC.TraversalSpecsStylesheet
       (CREATE.STYLE 'ITEMS (LIST (create MENU ITEMS _ T))
              'SELECTIONS
              '(T)
              'ITEM.TITLES
              '(Forward% Links Backward% Links Depth)
              'ITEM.TITLE.FONT
              (FONTCOPY MENUFONT 'WEIGHT 'BOLD)
              'NEED.NOT.FILL.IN
              '(MULTI MULTI NIL)
              'TITLE "Include cards at:"))

(DEFINEQ
```

(NC.AskTraversalSpecs

```
  [LAMBDA (NoteFile OldLinkLabels OldDepth Don'tAskFlg InterestedWindow)
                                                          (* kirk%: " 5-May-87 14:15")
```

(* * Get a traversal specification from the user.)

(* * kirk |7/29/86| changed to allow backlinks and position specs above source card)

(* * rht 8/29/86%: Fixed bug that was causing Depth spec to be ignored.)

(* * rht 3/9/87%: Now accepts InterestedWindow argument. Now takes NoteFile rather than SourceCard arg.)

(* * rht 3/25/87%: Now calls NC.CoerceToInterestedWindow.)

(* * kirk 5/5/87%: Removed failed attempt at putting up hourglass)

```
    (DECLARE (GLOBALVARS NC.TraversalSpecsStylesheet))
    (OR (OPENWP InterestedWindow)
        (SETQ InterestedWindow (NC.CoerceToInterestedWindow NoteFile)))
    (PROG ((LinkLabels (NC.RetrieveLinkLabels NoteFile T))
           Choices Position)
          (OR OldLinkLabels (SETQ OldLinkLabels LinkLabels))
          (if Don'tAskFlg
              then (RETURN (LIST OldLinkLabels OldDepth)))
          [SETQ Position (AND (WINDOWP InterestedWindow)
                              (create POSITION
                                      XCOORD _ (fetch (REGION LEFT) of (WINDOWPROP InterestedWindow 'REGION))
                                      YCOORD _ (fetch (REGION TOP) of (WINDOWREGION InterestedWindow]
          (OR OldDepth (SETQ OldDepth 99999))

          (* The stylesheet is in a global var. We only need to provide its position, items, and selections.)

          (STYLE.PROP NC.TraversalSpecsStylesheet 'POSITION Position)
          [STYLE.PROP NC.TraversalSpecsStylesheet 'ITEMS
                 (LIST (create MENU
                               ITEMS _ LinkLabels)
                       (create MENU
                               ITEMS _ (for Link in LinkLabels collect (PACK* '_ Link)))
                       (create MENU
                               ITEMS _ '(0 1 2 3 4 5 6 7 8 9 INF]
```

```
          (STYLE.PROP NC.TraversalSpecsStylesheet 'SELECTIONS (LIST (for Label in OldLinkLabels
                                                                       when (NEQ (NTHCHAR Label 1)
                                                                                 '_)
                                                                       collect Label)
                                                                    (for Label in OldLinkLabels
                                                                       when (EQ (NTHCHAR Label 1)
                                                                                '_)
                                                                       collect Label)
                                                                    (if (OR (NOT (FIXP OldDepth))
                                                                            (IGREATERP OldDepth 9)
                                                                            (ILESSP OldDepth 0))
                                                                        then 'INF
                                                                      else OldDepth)))
          (SETQ Choices (STYLESHEET NC.TraversalSpecsStylesheet))
          (RETURN (COND
                     (Choices (create TRAVERSALSPECS
                                      LinkTypes _ (APPEND (CAR Choices)
                                                          (CADR Choices))
                                      Depth _ (OR (FIXP (CADDR Choices))
                                                  MAX.FIXP)))
                     (T NIL])

)


;;; UIDs

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.UIDBasis NC.GlobalUIDHashArray)
)

(RPAQ? NC.UIDBasis NIL)

(DEFINEQ
```

(NC.MakeUID

```
  [LAMBDA NIL                                                    (* Randy.Gobbel " 5-Nov-86 16:30")
```

(* * Return a new unique identifier composed of the NS host number, a logon time stamp, and a global counter since login.)

```
    (DECLARE (GLOBALVARS NC.UIDBasis))
    (COND
       ((OR (NULL NC.UIDCtr)
            (NULL NC.UIDBasis))
        (NC.InitializeUID)))
    (SETQ NC.UIDCtr (ADD1 NC.UIDCtr))
    (create UID copying NC.UIDBasis UID5 _ [LOGAND (LRSH NC.UIDCtr 16)
                                                   (CONSTANT (SUB1 (EXPT 2 16]
                      UID6 _ (LOGAND NC.UIDCtr (CONSTANT (SUB1 (EXPT 2 16])
```

(NC.LookUpUIDInHashArray

```
  [LAMBDA (UID)                                                  ; Edited 11-May-88 21:04 by Trigg
```

;; Looks up UID in a global hash array and returns existing object if there is one, else stashes this one in array.

```
    (DECLARE (GLOBALVARS NC.GlobalUIDHashArray))
    (if (OR (NOT (BOUNDP 'NC.GlobalUIDHashArray))
            (NOT (HASHARRAYP NC.GlobalUIDHashArray)))
        then (SETQ NC.GlobalUIDHashArray (NC.CreateUIDHashArray 1000)))
    (if (GETHASH UID NC.GlobalUIDHashArray)
      else (PUTHASH UID UID NC.GlobalUIDHashArray)
           UID])
```

(NC.InitializeUID

```
  [LAMBDA NIL                                                    (* Randy.Gobbel " 5-Nov-86 16:30")
```

(* * Initialize the UID mechanism)

```
    (DECLARE (GLOBALVARS \MY.NSHOSTNUMBER NC.UIDBasis))
    (LET ((IDate (IDATE)))
         (SETQ NC.UIDCtr 1)
         (SETQ NC.UIDBasis (create UID
                                   UID0 _ (CADR \MY.NSHOSTNUMBER)
                                   UID1 _ (CADDR \MY.NSHOSTNUMBER)
                                   UID2 _ (CADDDR \MY.NSHOSTNUMBER)
                                   UID3 _ [LOGAND (LRSH IDate 16)
                                                  (CONSTANT (SUB1 (EXPT 2 16]
                                   UID4 _ [LOGAND IDate (CONSTANT (SUB1 (EXPT 2 16]
                                   UID5 _ 0
                                   UID6 _ 0))
         (ADDTOVAR \SYSTEMCACHEVARS NC.UIDBasis NC.UIDCtr)])
```

(NC.SameUIDP

```
  [LAMBDA (UID1 UID2)                                            (* fgh%: "20-Nov-85 18:51")
```

```
          (* * Return non-nil if UIDs are the same.)

    (AND (type? UID UID1)
         (type? UID UID2)
         (EQP (ffetch (UID UID0) of UID1)
              (ffetch (UID UID0) of UID2))
         (EQP (ffetch (UID UID1) of UID1)
              (ffetch (UID UID1) of UID2))
         (EQP (ffetch (UID UID2) of UID1)
              (ffetch (UID UID2) of UID2))
         (EQP (ffetch (UID UID3) of UID1)
              (ffetch (UID UID3) of UID2))
         (EQP (ffetch (UID UID4) of UID1)
              (ffetch (UID UID4) of UID2))
         (EQP (ffetch (UID UID5) of UID1)
              (ffetch (UID UID5) of UID2))
         (EQP (ffetch (UID UID6) of UID1)
              (ffetch (UID UID6) of UID2])

)

(DEFINEQ

(NC.UIDPutProp
  [LAMBDA (UID Prop Value)                                    (* rht%: "26-Nov-85 22:25")
    (LET ((PropList (fetch (UID UserData) of UID)))
         (COND
            (PropList (LISTPUT PropList Prop Value))
            (T (replace (UID UserData) of UID with (LIST Prop Value])

(NC.UIDGetProp
  [LAMBDA (UID Prop)                                          (* fgh%: "20-Nov-85 19:10")
    (LISTGET (fetch (UID UserData) of UID)
           Prop])

(NC.UIDSetPropList
  [LAMBDA (UID PropList)                                      (* rht%: "25-Nov-85 23:38")
    (replace (UID UserData) of UID with PropList])

(NC.UIDGetPropList
  [LAMBDA (UID)                                               (* rht%: " 1-Feb-86 14:27")

          (* * Return the user data field of UID.)

    (fetch (UID UserData) of UID])

(NC.UIDAddProp
  [LAMBDA (UID Prop New Flg)                                  (* rht%: "26-Nov-85 00:12")
    (LET ((CurrentPropValue (NC.UIDGetProp UID Prop)))
         (if (LISTP CurrentPropValue)
             then (NC.UIDPutProp UID Prop (if Flg
                                              then (CONS New CurrentPropValue)
                                            else (NCONC1 CurrentPropValue New)))
           else (NC.UIDPutProp UID Prop (LIST New])

(NC.UIDRemProp
  [LAMBDA (UID Prop)                                          (* fgh%: "20-Nov-85 19:10")
    (NC.UIDPutProp UID Prop NIL])

)

;;; This stuff makes it possible for UIDs encountered by PRINT when writing card's proplists, to be written down in a way that can be read back by READ.

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.VerticalBarREADTABLE)
)

(DEFINEQ

(NC.BuildVerticalBarREADTABLE
  [LAMBDA NIL                                                 (* rht%: " 1-Nov-86 14:52")

          (* * Return a new readtable formed from the primary readtable by changing verticalbar's syntax class to be that of terminal
          readtable.)

          (* * rht 11/1/86%: Now makes copy of ORIG readtable rather than "NIL" one.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((NewReadTable (COPYREADTABLE NC.OrigReadTable)))
```

```
         (SETSYNTAX 124 (GETSYNTAX 124 T)
                NewReadTable)
         NewReadTable])

)

(RPAQ? NC.VerticalBarREADTABLE (NC.BuildVerticalBarREADTABLE))

(DEFINEQ
```

(NC.ReassembleUID

```
  [LAMBDA (UID0 UID1 UID2 UID3 UID4 UID5 UID6)                    ; Edited 11-May-88 21:04 by Trigg
```

;;; Called when a UID that was written down by PRINT after disassembling so can be reassembled into a UID.

;; rht 5/11/88: Now calls NC.LookUpUIDInHashArray so can return prior UID object if we've seen it before.

```
    (NC.LookUpUIDInHashArray (create UID
                                     UID0 _ UID0
                                     UID1 _ UID1
                                     UID2 _ UID2
                                     UID3 _ UID3
                                     UID4 _ UID4
                                     UID5 _ UID5
                                     UID6 _ UID6])
```

(NC.DisassembleUID

```
  [LAMBDA (UID Stream)                                            (* rht%: " 1-Nov-86 17:34")
```

(* * This is called when PRINT, PRIN1, etc. try to print instance of UID datatype.
Returns cons of readmacro character and list to be written down which when eval'ed at read time will reassemble the UID.)

(* * rht 1/31/86%: Now checks Stream arg and only disassembles if going to a notecards stream.)

(* * rht 2/1/86%: It's too slow to check whether Stream is for a notefile.
Now just check whether there's a non-nil and non-T Stream.)

(* * rht 2/4/86%: Also be sure Stream is not a window stream.)

(* * rht 11/1/86%: Changed a little weirdness in the readmacro atom so compiled version of this fn can be read in lyric.)

```
    (if (AND Stream (NEQ Stream 'T)
             (NOT (IMAGESTREAMP Stream)))
        then (CONS (MKATOM "|.")
                   (LIST (FUNCTION NC.ReassembleUID)
                         (ffetch (UID UID0) of UID)
                         (ffetch (UID UID1) of UID)
                         (ffetch (UID UID2) of UID)
                         (ffetch (UID UID3) of UID)
                         (ffetch (UID UID4) of UID)
                         (ffetch (UID UID5) of UID)
                         (ffetch (UID UID6) of UID)))
      else NIL])

)

(DECLARE%: DONTEVAL@LOAD

(DEFPRINT 'UID (FUNCTION NC.DisassembleUID))
)
```

;;; contention lock machinery

```
(DEFINEQ
```

(NC.NoteFileProp

```
  [LAMBDA X                                                       (* fgh%: "27-Jun-86 17:06")
```

(* * Set or fetch a user data prop from a NF.)

(* * fgh |6/25/86| First created.)

```
    (LET [(UserDataPropList (fetch (NoteFile UserProps) of (ARG X 1]
         (if (GREATERP X 2)
             then [COND
                     ((LISTP UserDataPropList)
                      (LISTPUT UserDataPropList (ARG X 2)
                             (ARG X 3)))
                     (T (replace (NoteFile UserProps) of (ARG X 1) with (LIST (ARG X 2)
                                                                              (ARG X 3]
           else (AND (LISTP UserDataPropList)
                     (LISTGET UserDataPropList (ARG X 2])
```

(NC.NoteFileAddProp

```
  [LAMBDA (NoteFile PROP NEWVALUE)                                (* fgh%: " 3-Sep-86 01:38")
```

(* * Includes NEWVALUE in the LIST of values for the PROP property of the UserProps list for NoteFile.)

```
    (PROG ((CURRENTPROPVALUELIST (NC.NoteFileProp NoteFile PROP)))
          (RETURN (COND
                      ((LISTP CURRENTPROPVALUELIST)
                       (NC.NoteFileProp NoteFile PROP (CONS NEWVALUE CURRENTPROPVALUELIST)))
                      (T (NC.NoteFileProp NoteFile PROP (LIST NEWVALUE])
```

(NC.NoteFileDelProp

```
  [LAMBDA (NoteFile PROP VALUE MATCHFN)                    (* fgh%: " 3-Sep-86 02:10")
```

(* * Opposite of NCSERVER.NoteFileAddProp. If VALUE is present on the list of values for PROP, then it is removed. MATCHFN if provided determines when VALUE matches some element from the list of values to be removed. Default MATCHFN is EQ.)

```
    (PROG [(PROPVALUES (NC.NoteFileProp NoteFile PROP))
           (REALMATCHFN (OR MATCHFN (FUNCTION EQ]
          (RETURN (NC.NoteFileProp NoteFile PROP (DREMOVE (find Val in PROPVALUES
                                                                suchthat (APPLY* REALMATCHFN Val VALUE))
                                                          PROPVALUES])

)

(DEFINEQ
```

(NC.PrintOperationInProgressMsg

```
  [LAMBDA (Window Operation OperationInProgress)           (* Randy.Gobbel " 2-Apr-87 19:39")
```

(* * Notify the user that they are attempting to do an operation while another operation is in progress. Use a prompt window above the card's promptwindow because the card's prompt window is probably being used for the operation.)

(* * fgh |6/9/86| First created.)

(* * RG |3/10/87| removed redundant (?) call to NC.AttachPromptWindow)

(* * rg |4/2/87| rewritten to not use prompt window stuff)

```
    (LET* [(MainWindowRegion (WINDOWPROP Window 'REGION))
           (PWinWidth (MAX (WIDTHIFWINDOW (fetch (REGION WIDTH) of MainWindowRegion))
                           300))
           (PWindow (OR (for WCandidate in (ALLATTACHEDWINDOWS Window) thereis (AND (WINDOWPROP WCandidate
                                                                                                 'NC.OpInProgressWindow)
                                                                                    (OPENWP WCandidate)))
                        (CREATEW [create REGION
                                         LEFT _ 0
                                         BOTTOM _ 0
                                         WIDTH _ PWinWidth
                                         HEIGHT _ (SETQ HEIGHT (HEIGHTIFWINDOW (TIMES 2 (FONTPROP (DSPFONT NIL
                                                                                                          Window)
                                                                                                  'HEIGHT]
                                 NIL NIL T]
          (WINDOWPROP PWindow 'NC.OpInProgressWindow T)
          [ATTACHWINDOW PWindow Window 'TOP (COND
                                              ((LEQ (PLUS (fetch (REGION LEFT) of MainWindowRegion)
                                                          PWinWidth)
                                                    SCREENWIDTH)
                                               'LEFT)
                                              ((LEQ (PLUS (fetch (REGION LEFT) of MainWindowRegion)
                                                          (LRSH (PLUS PWinWidth (fetch (REGION WIDTH)
                                                                                   of MainWindowRegion))
                                                                1))
                                                    SCREENWIDTH)
                                               'CENTER)
                                              (T 'RIGHT]
          (NC.PrintMsg PWindow T "## Can't " Operation (CHARACTER 13)
                 "## " OperationInProgress " in progress.")
          (REPOSITIONATTACHEDWINDOWS Window)
          (DISMISS 2000)
          (REMOVEWINDOW PWindow])
```

(NC.CardOperationsInProgress

```
  [LAMBDA (NoteFile AskAndTerminateFlg)                    (* Randy.Gobbel " 5-Mar-87 15:41")
```

(* * Check if any cards have operations in progress. If so, either return their processes or Ask the user if they should be terminated and termionate them, depending on AskAndTerminateFlg.
If the user say no terminations, then return the LITATOM ABORT.)

(* * fgh |6/13/86| First created)

(* * fgh |7/16/86| Due to change of NC.MapCards from fn to macro had to change the RETFROM NC.MapVCards to a RETFROM MAPHASH.)

(* * rg |3/4/87| rewritten to sort of cooperate with new concurrency machinery.
WARNING%: THE VALUE RETURNED FROM THIS FUNCTION IS VOLATILE AND SHOULD NOT BE RELIED ON.

IN ADDITION, CALLING THIS FUNCTION MAY RESULT IN THE NOTECARDS WORLD BEING LEFT IN AN INCONSISTENT STATE)

```
(WITH.MONITOR NC.LockLock
    [LET (Processes (NC.NoteFileProp NoteFile 'CardProcessInProgressList))
         (if (NULL AskAndTerminateFlg)
             then                                                    (* Just Return a lst of proceses)
                  Processes
           else (if (NULL Processes)
                    then NIL
                  else (if (NC.AskYesOrNo (CONCAT "There are cards with operations in progress." (CHARACTER
                                                                                                      13)
                                                  "Do you want to terminate these operations?"
                                                  (CHARACTER 13))
                                           "-->"
                                           'Yes T (WFROMMENU (fetch (NoteFile Menu) of NoteFile)))
                           then [for Process in Processes do (PROCESS.EVAL Process '(ERROR!]
                         else 'ABORT])])
```

(NC.OperationInProgress

```
[LAMBDA (Card)                                                       (* fgh%: "25-Jun-86 19:55")
```

(* * Return the process for the operation in prgress on Card, if any)

(* * fgh |6/13/86| First created.)

```
(OR (AND (PROCESSP (NC.FetchUserDataProp Card 'ProcessInProgress))
         (NC.FetchUserDataProp Card 'OperationInProgress))
    (AND (PROCESSP (NC.NoteFileProp (fetch (Card NoteFile) of Card)
                                    'ProcessInProgress))
         (NC.NoteFileProp (fetch (Card NoteFile) of Card)
                          'OperationInProgress])
```

(NC.CardCheckOpInProgress

```
[LAMBDA (Card)                                                       (* Randy.Gobbel " 6-Mar-87 12:04")
```

(* * Return the operation in progress if any. Checks session level, then NoteFile level, then Card level.
Returns NIL if there is no conflict at any level, otherwise a string describing the conflicting operation.
THIS PROCEDURE IS INTERNAL TO THE NOTECARDS GLOBAL MONITOR!)

(* * rg |3/3/87| First created.)

```
(LET (ProcInProgress)
     (COND
        [(PROCESSP (SETQ ProcInProgress NC.SessionProcessInProgress))
                                                                     (* someone doing a session op, check to see if it's us)
         (COND
            ((NEQ ProcInProgress (THIS.PROCESS))                     (* if someone else has the session lock, return that op)
             NC.SessionOperationInProgress)
            (T                                                       (* we have the session lock)
               'US]
        [[PROCESSP (SETQ ProcInProgress (NC.NoteFileProp (fetch (Card NoteFile) of Card)
                                                         'ProcessInProgress]
                                                                     (* someone doing a NF op, check to see if it's us)
         (COND
            ((NEQ ProcInProgress (THIS.PROCESS))                     (* if someone else has the NF lock, return that op)
             (NC.NoteFileProp (fetch (Card NoteFile) of Card)
                              'OperationInProgress))
            (T                                                       (* we have the NF lock)
               'US]
        [[PROCESSP (SETQ ProcInProgress (NC.FetchUserDataProp Card 'ProcessInProgress]
                                                                     (* if someone has the card lock, see if it's us)
         (COND
            ((NEQ ProcInProgress (THIS.PROCESS))                     (* if someone else has the card lock, return that op)
             (NC.FetchUserDataProp Card 'OperationInProgress))
            (T                                                       (* we have the card lock)
               'US]
        (T                                                           (* all relevant locks are free)
           NIL])
```

(NC.NoteFileCheckOpInProgress

```
[LAMBDA (NoteFile)                                                   (* Randy.Gobbel " 5-Jun-87 14:26")
```

(* * NoteFile level check for operation in progress. Checks session level, then NoteFile level, then Card level.
Returns NIL if there is no conflict at any level, otherwise a string describing the conflicting operation.
THIS PROCEDURE IS INTERNAL TO THE NOTECARDS GLOBAL MONITOR!)

(* * rg |3/3/87| created)

(* * rg |6/2/87| changed to allow enlarging lock scope if it's all in the same process)

```
(LET (ProcInProgress)
     (COND
        ((PROCESSP NC.SessionProcessInProgress)                      (* someone doing a session op, check to see if it's us)
```

```
            (COND
               ((NEQ NC.SessionProcessInProgress (THIS.PROCESS))
                                                          (* if someone else has the session lock, return that op)
                NC.SessionOperationInProgress)
               (T                                         (* we have the session lock, just return NIL)
                  NIL)))
            ([PROCESSP (SETQ ProcInProgress (NC.NoteFileProp NoteFile 'ProcessInProgress]
                                                          (* someone doing a NF op, check to see if it's us)
             (COND
                ((NEQ ProcInProgress (THIS.PROCESS))      (* if someone else has the NF lock, return that op)
                 (NC.NoteFileProp NoteFile 'OperationInProgress))
                (T                                        (* we have the NF lock, do nothing)
                   NIL)))
            ((for Process in (NC.NoteFileProp NoteFile 'CardProcessInProgressList) always (EQ Process (THIS.PROCESS)
                                                                                           ))
                                                          (* card ops in progress by our own process)
             NIL)
            ((NC.NoteFileProp NoteFile 'CardProcessInProgressList)(* some card ops in progress, we lose)
             "Card Operations")
            (T                                            (* all relevant locks are free)
               NIL])
```

(NC.SessionCheckOpInProgress

```
  [LAMBDA NIL                                             (* Randy.Gobbel " 5-Jun-87 14:25")
```

(* * Session level check for operation in progress. Checks session level, then NoteFile level, then Card level. Returns NIL if there is no conflict at any level, otherwise a string describing the conflicting operation. THIS PROCEDURE IS INTERNAL TO THE NOTECARDS GLOBAL MONITOR!)

(* * rg |3/3/87| created)

(* * rg |6/2/87| changed to allow enlarging lock scope if it's all in the same process)

```
    (COND
       [(PROCESSP NC.SessionProcessInProgress)            (* someone doing a session op, check to see if it's us)
        (COND
           ((NEQ NC.SessionProcessInProgress (THIS.PROCESS))   (* if someone else has the session lock, return that op)
            NC.SessionOperationInProgress)
           (T                                             (* we have the session lock)
              'US]
       ((for Process in NC.NoteFileBusyList always (EQ Process (THIS.PROCESS)))
                                                          (* notefile ops in progress by our own process, assume we know
        what we're doing)
        NIL)
       (NC.NoteFileBusyList                               (* NF ops are active, we lose)
             "NoteFile Operations")
       ((for Process in NC.CardBusyList always (EQ Process (THIS.PROCESS)))
                                                          (* card ops in progress, but it's our own process)
        NIL)
       (NC.CardBusyList                                   (* some card ops are active, we lose)
             "Card Operations")
       (T                                                 (* all relevant locks are free)
          NIL])
```

(NC.SessionToNoteFileLock

```
  [LAMBDA (NoteFile Operation)                            ; Edited  3-Dec-87 18:59 by rht:
```

(* * downgrade session lock to NoteFile lock for specified NF. MUST BE CALLED ONLY FROM WITHIN NC.ProtectedSessionOperation!)

(* * rg |3/9/87| created)

```
    (COND
       ((NEQ NC.SessionProcessInProgress (THIS.PROCESS))
        (NC.ReportError "NC.SessionToNoteFileLock" "Attempted to release session lock when not held by this
              process")))
    (OBTAIN.MONITORLOCK NC.LockLock)
    [RESETSAVE (NC.NoteFileProp NoteFile 'OperationInProgress Operation)
           `(NC.NoteFileProp ,NoteFile OperationInProgress ,(NC.NoteFileProp NoteFile 'OperationInProgress]
    [RESETSAVE (NC.NoteFileProp NoteFile 'ProcessInProgress (THIS.PROCESS))
           `(NC.NoteFileProp ,NoteFile ProcessInProgress ,(NC.NoteFileProp NoteFile 'ProcessInProgress]
    [RESETSAVE (SETQ NC.NoteFileBusyList (CONS (THIS.PROCESS)
                                               NC.NoteFileBusyList))
           '(SETQ NC.NoteFileBusyList (DREMOVE (THIS.PROCESS)
                                               NC.NoteFileBusyList]
    (SETTOPVAL 'NC.SessionProcessInProgress NIL)
    (SETTOPVAL 'NC.SessionOperationInProgress NIL)
    (RELEASE.MONITORLOCK NC.LockLock])
```

(NC.LockListOfCards

```
  [LAMBDA (CardIdentifiers Operation FileLevelFlg)        ; Edited  3-Dec-87 18:59 by rht:
```

(* * set locks on cards passed in, return a list of the lock statuses. Status = NIL means lock was free, = (QUOTE US) means we already had it, = <string> means operation described by the string was already in progress on that card)

```
        (* * RG |4/2/87| created)

        (* * rg |6/2/87| added FileLevelFlg)

    (DECLARE (USEDFREE CardListResetVar))
    (WITH.MONITOR NC.LockLock
        (LET* [(Cards (for CardIdentifier in CardIdentifiers collect (NC.CoerceToCard CardIdentifier)))
               (LockStatusList (for Card in Cards collect (if FileLevelFlg
                                                              then (NC.NoteFileCheckOpInProgress (fetch (Card NoteFile)
                                                                                                        of Card))
                                                            else (NC.CardCheckOpInProgress Card]
              [for Card in Cards as Status in LockStatusList when (NULL Status)
                 do (if FileLevelFlg
                        then [NAMED-RESETSAVE CardListResetVar (NC.NoteFileProp (fetch (Card NoteFile) of Card)
                                                                                'OperationInProgress Operation)
                                              `(NC.NoteFileProp ,(fetch (Card NoteFile) of Card)
                                                     OperationInProgress
                                                     ,(NC.NoteFileProp NoteFile 'OperationInProgress]
                             [NAMED-RESETSAVE CardListResetVar (NC.NoteFileProp (fetch (Card NoteFile) of Card)
                                                                                'ProcessInProgress
                                                                                (THIS.PROCESS))
                                              `(NC.NoteFileProp ,(fetch (Card NoteFile) of Card)
                                                     ProcessInProgress
                                                     ,(NC.NoteFileProp (fetch (Card NoteFile) of Card)
                                                            'ProcessInProgress]
                             [NAMED-RESETSAVE CardListResetVar (SETQ NC.NoteFileBusyList (CONS (THIS.PROCESS)

                                                                                            NC.NoteFileBusyList
                                                                                             ))
                                              '(SETQ NC.NoteFileBusyList (DREMOVE (THIS.PROCESS)
                                                                                  NC.NoteFileBusyList]
                          else [NAMED-RESETSAVE CardListResetVar (NC.SetUserDataProp Card 'ProcessInProgress (
                                                                                                THIS.PROCESS
                                                                                                    ))
                                               `(NC.SetUserDataProp ,Card ProcessInProgress ,(NC.FetchUserDataProp
                                                                                              Card
                                                                                              'ProcessInProgress]
                                  [NAMED-RESETSAVE CardListResetVar (NC.SetUserDataProp Card 'OperationInProgress
                                                                                        Operation)
                                               `(NC.SetUserDataProp ,Card OperationInProgress ,(NC.FetchUserDataProp
                                                                                                Card
                                                                                                'OperationInProgress]
                                  [NAMED-RESETSAVE CardListResetVar [NC.NoteFileProp (fetch (Card NoteFile) of Card)
                                                                                     'CardProcessInProgressList
                                                                                     (CONS (THIS.PROCESS)
                                                                                           (NC.NoteFileProp (fetch (Card NoteFile)
                                                                                                                   of Card)
                                                                                                            'CardProcessInProgressList]
                                               `(NC.ResetCardProcessInProgress ,(fetch (Card NoteFile) of Card]
                                  (NAMED-RESETSAVE CardListResetVar (SETQ NC.CardBusyList (CONS (THIS.PROCESS)
                                                                                                NC.CardBusyList))
                                               '(SETQ NC.CardBusyList (DREMOVE (THIS.PROCESS)
                                                                               NC.CardBusyList]
             LockStatusList))])
```

(NC.ResetCardProcessInProgress

```
  [LAMBDA (NoteFile)                                                    (* Randy.Gobbel " 5-Mar-87 15:31")
    (NC.NoteFileProp NoteFile 'CardProcessInProgressList (DREMOVE (THIS.PROCESS)
                                                                  (NC.NoteFileProp NoteFile
                                                                                   'CardProcessInProgressList])
```

(NC.SwitchNoteFileLock

```
  [LAMBDA (NoteFile OldProcess)                                         (* Randy.Gobbel "14-Jul-87 19:11")

        (* * rg |6/26/87| Switch lock for this notefile from passed-in process to current process.
        WARNING%: USE WITH CAUTION. IF YOU DON'T KNOW WHAT YOU'RE DOING, DON'T CALL THIS FUNCTION)

    (DECLARE (GLOBALVARS NC.NoteFileBusyList))
    (UNINTERRUPTABLY
        (NC.NoteFileProp NoteFile 'ProcessInProgress (THIS.PROCESS))
        (DSUBST (THIS.PROCESS)
               OldProcess RESETVARSLST)
        (DSUBST (THIS.PROCESS)
               OldProcess NC.NoteFileBusyList)
        (THIS.PROCESS))])

)

;;; Miscellaneous.

(DEFINEQ

(NC.GetNewCard
```

```
  [LAMBDA (NoteFile Type OverrideUID)                    ; Edited 22-Dec-88 17:37 by sye

          (* * See NCLocalDevice.NewCardUID.)

          (* * kef 7/17/86%: Updated to use the device vector function to grab an IndexLoc in the local case, and the UID in all cases.)

          (* * kef 8/4/86%: Now takes Type argument and passes it onto the NewCardUIDFn)

          (* * fgh |8/31/86| Changed APPLY* to NC.ApplyFn.)

          (* * rht 1/22/87%: Added OverrideUID argument. Use with GREAT care!)

    (LET ((Card (create Card
                        NoteFile _ NoteFile
                        UID _ OverrideUID))
          ReturnValue)
         (COND
            ((type? Card (SETQ ReturnValue (NC.ApplyFn NewCardUIDFn Card Type)))
             (NC.InstallCardInNoteFile Card NoteFile)
             (NC.DisplayFileCapacity NoteFile NIL 'NC.GetNewCard))
            (T (NC.ReportError 'NC.GetNewCard ReturnValue)))
         Card])
```

(NC.DatabaseFileName

```
  [LAMBDA (Msg Prompt ClearFirstFlg NoSuggestFlg Name InterestedWindow)
                                                         ; Edited  8-Dec-88 16:48 by krivacic
    ;; Make a NoteCards database file name on the base specified by the user.  Basically, add the NOTEFILE extension

;;; rht 8/7/84: Now provides file name suggestion for user (unless NoSuggestFlg is non-nil.) The suggestion is in the global var
;;; NC.DatabaseFileNameSuggestion which is reset to the new file name before returning.

;;; kirk 23Jan86 Added optional InterestedWindow

;;; rht 7/2/86: Fixed to just pass InterestedWindow to NC.AskUser rather than computing the prompt window of that.

;;; kef 7/21/86: Removed the call to FULLNAME, which was causing problems if the file happened to be located on a NoteCards server that did not also
;;; support FileService.

;;; fgh 8/31/86 Restored FULLNAME for Non-remote hosts.

    (DECLARE (GLOBALVARS NC.DatabaseFileNameSuggestion))
    [OR Name (SETQ Name (MKATOM (NC.AskUser Msg Prompt (AND (NOT NoSuggestFlg)
                                                            NC.DatabaseFileNameSuggestion)
                                       ClearFirstFlg InterestedWindow]
    (if Name
        then (SETQ Name (PACKFILENAME 'BODY Name 'EXTENSION 'NOTEFILE))
             [SETQ NC.DatabaseFileNameSuggestion (PACKFILENAME 'VERSION NIL 'BODY (if (NC.RemoteHostP Name)
                                                                                     then Name
                                                                                   else (FULLNAME Name]
             Name
      else NIL])
```

(NC.WriteStatus

```
  [LAMBDA (Stream Status)                                (* rht%: "15-Nov-85 18:18")

          (* * Write a 1 byte status to stream)

    (BOUT Stream (CHCON1 (SUBATOM Status 1 1])
```

(NC.TotalCardsInNoteFile

```
  [LAMBDA (NoteFile)                                     (* rht%: "14-Mar-87 01:11")

          (* * Return the total number of cards in NoteFile. NoteFile should be open.)

    (if (NC.NoteFileOpenP NoteFile)
        then (DIFFERENCE (SUB1 (fetch (NoteFile NextIndexNum) of NoteFile))
                         (LENGTH (fetch (NoteFile IndexNumsFreeList) of NoteFile])

)

(ADDTOVAR HPRINTMACROS (FONTDESCRIPTOR . WRITE.FONTDESCRIPTOR))

(DEFINEQ
```

(WRITE.FONTDESCRIPTOR

```
  [LAMBDA (FONTDESCRIPTOR OUTFILE)                       ; Edited  3-Dec-87 18:59 by rht:
                                                         (* writes out the name of a font instead of the descriptor.)
                                                         (* only works for TEXTSTREAMS)

    (PRIN1 '(READ.FONTINTODESCRIPTOR)
           OUTFILE)
    (printout OUTFILE "(" (FONTPROP FONTDESCRIPTOR 'FAMILY)
           %,
           (FONTPROP FONTDESCRIPTOR 'SIZE)
           %,
```

```
            (FONTPROP FONTDESCRIPTOR 'FACE)
            %,
            (FONTPROP FONTDESCRIPTOR 'ROTATION)
            %,
            (FONTPROP FONTDESCRIPTOR 'DEVICE)
            ")" T)
    T])
```

(READ.FONTINTODESCRIPTOR

```
  [LAMBDA (FILE)                                          (* rrb " 4-OCT-83 19:06")
```

(* reads a text stream from the file that was written by WRITE.TEXTSTREAM which is an HPRINT macro.)

```
    (APPLY* (FUNCTION FONTCREATE)
           (READ FILE])

)

(DECLARE%: DONTEVAL@LOAD

(NC.StoreAutoloadFnFile (FUNCTION NC.FindNextCardPart)
       'NCREPAIR
       'NOTECARDSDIRECTORIES)
)

(PUTPROPS NCDATABASE FILETYPE :FAKE-COMPILE-FILE)

(PUTPROPS NCDATABASE MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA NC.NoteFileProp)
)

(PUTPROPS NCDATABASE COPYRIGHT ("Venue & Xerox Corporation" 1984 1985 1986 1987 1988 1989 1990 1993 1994 2021
                                ))
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX