File created: 5-Nov-2020 19:15:21 **{DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCDECLS.;10**

previous date: 3-Nov-2020 16:10:54 {DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCDECLS.;9

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ NCDECLSCOMS
       [
```

;;; LOAD THIS FILE BEFORE TRYING TO COMPILE ANY NOTECARDS FILES!!

;; Some definers for Notecards

;; WARNING: INITADVISE does not work for (ADVISE (MUMBLE IN MUMBLE)), which is a whole different animal.

```
(FILEPKGCOMS INITPROPS INITADVISE)
(FNS LOADINITPROPS LOADINITADVISE)
```

;; used in NCCONFIG

```
(FUNCTIONS NC.AutoloadApply NC.AutoloadApply* NAMED-RESETLST .NAMED-RESETLST. NAMED-RESETSAVE
       NAMED-RESETUNSAVE)
(MACROS NAMED-RESETLST .NAMED-RESETLST.)
(FNS NAMED-RESETUNWIND)
```

;; used in NCUTILITIES

```
(FUNCTIONS NC.WithTopWindowsUnattached ABORT.PROTECT)
(FUNCTIONS NC.ActivateCardAndDo)
```

;; used in NCPARAMETERS

```
(RECORDS GLOBALPARAMETER)
```

;; used in NCDATABASE

```
(RECORDS NoteFileDevice NoteFileCriticalUIDs NoteFile NoteFileVersion IndexLocs CardPartRecord
       TRAVERSALSPECS UID WORD)
                                                          ; akw removed MONITORLOCK
(FUNCTIONS NC.ProtectedCardOperation NC.ProtectedNoteFileOperation NC.ProtectedSessionOperation
       NC.IfAllCardsFree)
(FUNCTIONS NC.DoCardPartFn)
(FUNCTIONS NC.ReadPtr NC.WritePtr NC.ReadStatus)
                                                          ; The following aren't called anymore, but are saved for
                                                          ; reference purposes.
(MACROS NC.GetPtr NC.PutPtr NC.GetStatus NC.PutStatus)
```

;; used in NCCOMPACT

```
(RECORDS SortingRecord)
```

;; used in NCREPAIR

```
(RECORDS ScavengerInfo LINKSDATA TITLEDATA PROPLISTDATA)
```

;; used in NCCARDS

```
(RECORDS CardObject CardCache Card LinksCache)
(FUNCTIONS NC.WithWritePermission NC.IfCardPartNotBusy NC.IfMultipleCardPartsNotBusy)
                                                          ; Hash Array Handler
(FUNCTIONS NC.MapCards)
(RECORDS PropListItem)
```

;; used in NCLINKS

```
(RECORDS Link LINKDISPLAYMODE)
(RECORDS NCPointer)
```

;; used in NCTYPESMECH

```
(RECORDS NoteCardType)
(MACROS NC.ApplyFn NC.ApplySupersFn)
(FUNCTIONS NC.GetCardTypeField)
```

;; used in NCPROGINT

```
(FUNCTIONS NCP.ApplyCardTypeFn NCP.ApplySuperTypeFn)
(FUNCTIONS NCP.MapCards NCP.MapCardsOfType)
(FUNCTIONS NCP.WithLockedCards)
(RECORDS NOTECARDDATES)
(FUNCTIONS NCP.MapLinks NCP.MapLinksOfType)
```

;; used in NCCROSSFILELINKS

```
(RECORDS CrossFileLinkSubstance)
```

;; used in NCBROWSERCARD

```
(RECORDS SPECIALBROWSERSPECS)
```

```
        ;; used in NCCONVERTVERSION2TO3
        (RECORDS POINTERLIST NOTECARDLINK)
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              NCDECLS)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA LOADINITADVISE
                                                                                LOADINITPROPS)
                                                                           (NLAML)
                                                                           (LAMA])
```

;;; LOAD THIS FILE BEFORE TRYING TO COMPILE ANY NOTECARDS FILES!!

;; Some definers for Notecards

;; WARNING: INITADVISE does not work for (ADVISE (MUMBLE IN MUMBLE)), which is a whole different animal.

```
[PUTDEF 'INITPROPS 'FILEPKGCOMS '((COM MACRO (X (P (LOADINITPROPS . X]

[PUTDEF 'INITADVISE 'FILEPKGCOMS '((COM MACRO (X (P * (LIST (APPEND (LIST 'LOADINITADVISE (GENSYM 'NC-ADVICE))
                                                                    'X]

(DEFINEQ

(LOADINITPROPS
  [NLAMBDA spec-list                                          ; Edited  4-Dec-87 18:50 by rht:
    (for spec in spec-list do (LET ((var (CAR spec)))
                                (for prop-val in (CDR spec)
                                   do (LET ((prop (CAR prop-val))
                                            (val (CADR prop-val)))
                                           (if (NOT (GETPROP var prop))
                                               then (PUTPROP var prop val])

(LOADINITADVISE
  [NLAMBDA spec-list                                          ; Edited  5-Dec-87 12:57 by rht:
    (LET ((id (CAR spec-list)))
         (for spec in (CDR spec-list) do (LET ((fn (CAR spec))
                                               (advice (CADR spec)))
                                              (if (NOT (GETPROP fn id))
                                                  then (PUTPROP fn 'READVICE advice)
                                                       (APPLY* (FUNCTION READVISE)
                                                               fn)
                                                       (PUTPROP fn id T])

)

;; used in NCCONFIG

(DEFMACRO NC.AutoloadApply (Fn Args)

;;; Like NC.AutoloadApply* except takes single Args argument rather than spread.

    `[if (GETD ,Fn)
         then (APPLY ,Fn ,Args)
       else (LET [(FileName (NC.LookupAutoloadFnFile ,Fn]
                 (if FileName
                     then (NC.LoadFileFromDirectories FileName)
                          [if (GETD ,Fn)
                              then (APPLY ,Fn ,Args)
                            else (NC.ReportError "NC.AutoloadApply" (CONCAT "Loaded file " FileName ", but still no
                                                                            definition of " ,Fn]
                  else (NC.ReportError "NC.AutoloadApply" (CONCAT "No definition of " ,Fn " in table of
                                                                  autoloadable functions."])

(DEFMACRO NC.AutoloadApply* (Fn &REST args)

;;; First arg is name of function to apply. Rest are args. The fn name should be in the global table with an accompanying file to autoload from.

    `[if (GETD ,Fn)
         then (APPLY* ,Fn ,@args)
       else (LET [(FileName (NC.LookupAutoloadFnFile ,Fn]
                 (if FileName
                     then (NC.LoadFileFromDirectories FileName)
                          [if (GETD ,Fn)
                              then (APPLY* ,Fn ,@args)
                            else (NC.ReportError "NC.AutoloadApply*" (CONCAT "Loaded file " FileName ", but still
                                                                             no definition of " ,Fn]
                  else (NC.ReportError "NC.AutoloadApply*" (CONCAT "No definition of " ,Fn " in table of
                                                                   autoloadable functions."])

(DEFMACRO NAMED-RESETLST (HANDLE FORM)
   `[LET ((,HANDLE NIL)
          ((LISPXHIST LISPXHIST)))
         (DECLARE (SPECVARS ,HANDLE LISPXHIST))
```

```
         (CL:UNWIND-PROTECT
             ,FORM
             (NAMED-RESETUNWIND ,HANDLE))])

(DEFMACRO .NAMED-RESETLST. (HANDLE FORM &OPTIONAL INIT OTHERBINDINGS)
   `[LET ((,HANDLE ,INIT)
          ,@OTHERBINDINGS)
         [DECLARE (SPECVARS ,HANDLE ,@(MAPCAR OTHERBINDINGS 'CAR]
         (CL:UNWIND-PROTECT
             ,FORM
             (NAMED-RESETUNWIND ,HANDLE))])

(DEFMACRO NAMED-RESETSAVE (RESETHANDLE &REST FORMS)
   `(LET (RESETITEM)
         (SETQ ,RESETHANDLE (CONS [SETQ RESETITEM
                                  ,(COND
                                      [(AND (ATOM (CAR FORMS))
                                            (CAR FORMS))
                                       (SUBPAIR '(VAR VAL)
                                              (CDR X)
                                              '(PROG1 (CONS 'VAR (GETTOPVAL 'VAR))
                                                   (SETTOPVAL 'VAR VAL))]
                                      [(CDR FORMS)
                                       `(LIST ,(CADR FORMS)
                                              ,(CAR FORMS]
                                      (T `(LIST (LIST ',(COND
                                                          ((EQ (CAAR FORMS)
                                                               'SETQ)
                                                           (CAR (CADDAR FORMS)))
                                                          (T (CAAR FORMS)))
                                                      ,(CAR FORMS]
                                  ,RESETHANDLE))
         RESETITEM))

(DEFMACRO NAMED-RESETUNSAVE (RESETHANDLE ITEMFORM)
   `[LET ((ITEMLIST ,ITEMFORM))
         (for ITEM in ITEMLIST do (SET RESETHANDLE (DREMOVE ITEM ,RESETHANDLE))
                                  (COND
                                     ((LISTP (CAR ITEM))
                                      [SETQ OLDVALUE (COND
                                                         ((CDR ITEM)
                                                          (CADR ITEM))
                                                         (T (CADAR ITEM]
                                       (APPLY (CAAR ITEM)
                                              (CDAR ITEM)))
                                     (T (SETTOPVAL (CAR ITEM)
                                                   (CDR ITEM])

(DECLARE%: EVAL@COMPILE

[PROGN (DEFMACRO NAMED-RESETLST (HANDLE FORM)
          `[LET ((,HANDLE NIL)
                 ((LISPXHIST LISPXHIST)))
                (DECLARE (SPECVARS ,HANDLE LISPXHIST))
                (CL:UNWIND-PROTECT
                    ,FORM
                    (NAMED-RESETUNWIND ,HANDLE))])
       (PUTPROPS NAMED-RESETLST MACRO [(H X . Y)
                                       (.NAMED-RESETLST. H (PROGN X . Y)
                                              NIL
                                              ((LISPXHIST LISPXHIST])]

[PROGN (DEFMACRO .NAMED-RESETLST. (HANDLE FORM &OPTIONAL INIT OTHERBINDINGS)
          `[LET ((,HANDLE ,INIT)
                 ,@OTHERBINDINGS)
                [DECLARE (SPECVARS ,HANDLE ,@(MAPCAR OTHERBINDINGS 'CAR]
                (CL:UNWIND-PROTECT
                    ,FORM
                    (NAMED-RESETUNWIND ,HANDLE))])
       (PUTPROPS .NAMED-RESETLST. DMACRO (DEFMACRO (HANDLE FORM &OPTIONAL INIT OTHERBINDINGS) `[LET
                                                                                                ((,HANDLE
                                                                                                  ,INIT)

                                                                                                 ,@
                                                                                                  OTHERBINDINGS
                                                                                                 )
                                                                                                [DECLARE
                                                                                                 (SPECVARS
                                                                                                  ,HANDLE
                                                                                                  ,@(MAPCAR

                                                                                                   OTHERBINDINGS
                                                                                                      'CAR]
                                                                                                (CL:UNWIND-PRO
```

```
TECT,FORM
                                                                    (
                                                                NAMED-RESETUNWIND
                                                                    ,HANDLE))
                                                                ]))]
)

(DEFINEQ

(NAMED-RESETUNWIND
  [LAMBDA (HANDLE NORMALP)                                    ; Edited 29-Jan-88 18:14 by Randy.Gobbel
    ;; taken from Lyric RESETUNWIND of 4-Nov-86 16:53
    (while (LISTP HANDLE) bind OLDVALUE RESETZ do (SETQ RESETZ (pop HANDLE))
                                                  (if (LISTP (CAR RESETZ))
                                                      then          ; RESETSAVE and RESETFORM do this
                                                           (SETQ OLDVALUE (if (CDR RESETZ)
                                                                              then
                                                                              ;; occurs for RESETSAVE's when second argument is specified.  In this case, (CADR RESETZ) is the
                                                                              ;; value of the saving form, i.e. the first argument to RESETSAVE.
                                                                                  (CADR RESETZ)
                                                                              else (CADAR RESETZ)))
                                                                  (APPLY (CAAR RESETZ)
                                                                         (CDAR RESETZ))
                                                      else          ; RESETSAVE of a symbol sets its value
                                                           (SETTOPVAL (CAR RESETZ)
                                                                  (CDR RESETZ])

)

;; used in NCUTILITIES

(DEFMACRO NC.WithTopWindowsUnattached (MainWindow &BODY Forms)
    ;; Run Forms after temporarily detaching windows attached to top of MainWindow. This code was originally in Frank's NC.AttachNoteFileName
    ;; function, but found to be more generally useful.
    `(RESETLST
         [RESETSAVE NIL `(,[FUNCTION (LAMBDA (DescriptionList)    ; Reattach windows according to information in a description list.
                                     (for windowDescription in DescriptionList do (ATTACHWINDOW (CAR
                                                                                                    windowDescription
                                                                                                        )
                                                                                                ,MainWindow
                                                                                                (CAADR
                                                                                                    windowDescription
                                                                                                    )
                                                                                                (CDADR
                                                                                                    windowDescription
                                                                                                    ))
                                                                                            (WINDOWPROP (CAR
                                                                                                    windowDescription
                                                                                                       )
                                                                                                'PASSTOMAINCOMS
                                                                                                (CADDR
                                                                                                    windowDescription
                                                                                                    ]
                                 ,(bind WA for window in (ATTACHEDWINDOWS ,MainWindow)
                                     when [EQ 'TOP (CAR (SETQ WA (WINDOWPROP window 'WHEREATTACHED]
                                     collect
                                         ;; Detach attached windows at the top of the main window and retuirn a list describing their attachment.
                                         (PROG1 (LIST window WA (WINDOWPROP window 'PASSTOMAINCOMS))
                                                (DETACHWINDOW window]
         ,@Forms))

(DEFMACRO ABORT.PROTECT (MAIN-FORM CLEANUP-FORM)
    `(RESETLST
         [RESETSAVE NIL '(AND RESETSTATE ,CLEANUP-FORM]
         ,MAIN-FORM))

(DEFMACRO NC.ActivateCardAndDo (Card &REST Forms)

;;; rht 10/15/86: This written by MarkM. I changed slightly so as to accept a list of Forms rather than a singleton.

;;; rht&pmi 11/24: Changed name of localvar.

    `[LET [($$ActiveFlg$$ (NC.ActiveCardP ,Card]
          (OR $$ActiveFlg$$ (NC.GetNoteCard ,Card))
          (PROG1 (PROGN ,@Forms)
              (OR $$ActiveFlg$$ (NC.DeactivateCard ,Card)))])

;; used in NCPARAMETERS
```

```
(DECLARE%: EVAL@COMPILE

(RECORD GLOBALPARAMETER (PARAMGLOBALVAR PARAMFETCHFN PARAMSELECTIONFN PARAMCHECKFN PARAMAFTERCHANGEFN))
)
```

;; used in NCDATABASE

```
(DECLARE%: EVAL@COMPILE

(DATATYPE NoteFileDevice
       (ListNoteFilesFn CreateNoteFileFn DeleteNoteFileFn OpenNoteFileFn BuildHashArrayFn CloseNoteFileFn
              NoteFileOpenPFn CheckpointNoteFileFn NewCardUIDFn MarkCardDeletedFn GetCardInfoFn PutCardPartFn
              GetCardPartFn ObtainWritePermissionFn ReleaseWritePermissionFn CancelCacheSubscriptionFn
              RepairNoteFileFn CompactNoteFileFn TruncateNoteFileFn ConvertNoteFileFormatFn))

(RECORD NoteFileCriticalUIDs (NoteFile TableOfContents Orphans ToBeFiled LinkLabels Registry))

(DATATYPE NoteFile (UID Stream FullFileName HashArray (HashArraySize FIXP)
                        (NextIndexNum FIXP)
                        (Version BYTE)
                        (NextLinkNum FIXP)
                        (CheckptPtr FIXP)
                        LinkLabelsCard TableOfContentsCard ToBeFiledCard OrphansCard RegistryCard ReservedCards
                        Menu MonitorLock ExclusiveAccessMonitor CachingProcess IndexNumsFreeList UserProps
                        ReadOnlyFlg NoteFileDevice)
       (DATATYPE NoteFileDevice
              (ListNoteFilesFn CreateNoteFileFn DeleteNoteFileFn OpenNoteFileFn BuildHashArrayFn CloseNoteFileFn
                     NoteFileOpenPFn CheckpointNoteFileFn NewCardUIDFn MarkCardDeletedFn GetCardInfoFn
                     PutCardPartFn GetCardPartFn ObtainWritePermissionFn ReleaseWritePermissionFn
                     CancelCacheSubscriptionFn RepairNoteFileFn CompactNoteFileFn TruncateNoteFileFn
                     ConvertNoteFileFormatFn))
       (SYNONYM NoteFileDevice (Device)))

(TYPERECORD NoteFileVersion (Version NumberOfReservedCards NoteFileIndexWidth NoteFileHeaderSize))

(DATATYPE IndexLocs (MainCardDataLoc LinksLoc TitleLoc PropListLoc))

(RECORD CardPartRecord (FileLoc UID CardPartTypeNum CardPartLength))

(RECORD TRAVERSALSPECS (LinkTypes Depth))

(DATATYPE UID ((UID0 WORD)
               (UID1 WORD)
               (UID2 WORD)
               (UID3 WORD)
               (UID4 WORD)
               (UID5 WORD)
               (UID6 WORD)
               (UserData POINTER)))

(ACCESSFNS WORD ((HIBYTE (LRSH DATUM 8))
                 (LOBYTE (LOGAND DATUM 255)))
       (CREATE (IPLUS (LLSH HIBYTE 8)
                      LOBYTE)))
)

(/DECLAREDATATYPE 'NoteFileDevice
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '40)

(/DECLAREDATATYPE 'NoteFileDevice
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '40)

(/DECLAREDATATYPE 'NoteFile
       '(POINTER POINTER POINTER POINTER FIXP FIXP BYTE FIXP FIXP POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '46)

(/DECLAREDATATYPE 'IndexLocs '(POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '8)

(/DECLAREDATATYPE 'UID '(WORD WORD WORD WORD WORD WORD WORD POINTER)
       ;; ---field descriptor list elided by lister---
       '10)
```

```
;; akw removed MONITORLOCK

(DEFMACRO NC.ProtectedCardOperation (Card Operation InterestedWindow &BODY Body)
   `(RESETLST
        (OBTAIN.MONITORLOCK NC.LockLock)
        (LET [(OpInProgress (NC.CardCheckOpInProgress ,Card]
             [if (NULL OpInProgress)
                 then [RESETSAVE (NC.SetUserDataProp ,Card 'ProcessInProgress (THIS.PROCESS))
                             `(NC.SetUserDataProp ,,Card ProcessInProgress ,(NC.FetchUserDataProp ,Card
                                                                              'ProcessInProgress]
                      [RESETSAVE (NC.SetUserDataProp ,Card 'OperationInProgress ',Operation)
                             `(NC.SetUserDataProp ,,Card OperationInProgress ,(NC.FetchUserDataProp ,Card
                                                                                'OperationInProgress]
                      [RESETSAVE [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]
                      (RESETSAVE (SETQ NC.CardBusyList (CONS (THIS.PROCESS)
                                                             NC.CardBusyList))
                             '(SETQ NC.CardBusyList (DREMOVE (THIS.PROCESS)
                                                             NC.CardBusyList]
             (RELEASE.MONITORLOCK NC.LockLock)
             (if (OR (NULL OpInProgress)
                     (EQ OpInProgress 'US))
                 then ,@Body
               else (NC.PrintOperationInProgressMsg (OR ,InterestedWindow (NC.CoerceToInterestedWindow
                                                                           ,Card))
                          ,Operation OpInProgress)
                    'DON'T))))

(DEFMACRO NC.ProtectedNoteFileOperation (NoteFile Operation InterestedWindow &REST Body)
   `(RESETLST
        (OBTAIN.MONITORLOCK NC.LockLock)
        (LET [(OpInProgress (NC.NoteFileCheckOpInProgress ,NoteFile]
             [if (NULL OpInProgress)
                 then [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]
             (RELEASE.MONITORLOCK NC.LockLock)
             (if (OR (NULL OpInProgress)
                     (EQ OpInProgress 'US))
                 then ,@Body
               else (NC.PrintOperationInProgressMsg (OR (OPENWP ,InterestedWindow)
                                                        (NC.CoerceToInterestedWindow ,NoteFile))
                          ,Operation OpInProgress)
                    'DON'T))))

(DEFMACRO NC.ProtectedSessionOperation (Operation InterestedWindow &REST Body)
   `(RESETLST
        (OBTAIN.MONITORLOCK NC.LockLock)
        (LET ((OpInProgress (NC.SessionCheckOpInProgress)))
             (if (NULL OpInProgress)
                 then (RESETSAVE NC.SessionProcessInProgress (THIS.PROCESS))
                      (RESETSAVE NC.SessionOperationInProgress ,Operation))
             (RELEASE.MONITORLOCK NC.LockLock)
             (if (OR (NULL OpInProgress)
                     (EQ OpInProgress 'US))
                 then ,@Body
               else (NC.PrintOperationInProgressMsg ,InterestedWindow ,Operation OpInProgress)
                    'DON'T))))

(DEFMACRO NC.IfAllCardsFree (LockForm &BODY Body)
   [LET ((LockStatus (GENSYM)))
        `(LET ((,LockStatus ,LockForm))
              (if [for Status in ,LockStatus never (AND Status (NEQ Status 'US]
                  then ,@Body
                else ,LockStatus])

(DEFMACRO NC.DoCardPartFn (GetOrPut CardForm CardPartForm &REST Body)

;;; Call Get or Put card part fn both before and after Body.

;;; fgh 8/31/86 First created.
```

```
   [LET ((*READTABLE* (FIND-READTABLE "OLD-INTERLISP-T"))
         (*PACKAGE* *INTERLISP-PACKAGE*))
        `(RESETLST
             (RESETSAVE (NC.ApplyFn ,(PACK* GetOrPut 'CardPartFn)
                                ,CardForm
                                ,CardPartForm
                                'BEFORE)
                    `(APPLY* ,(fetch [Card ,(PACK* GetOrPut 'CardPartFn] of ,CardForm)
                           ,,CardForm
                           ,,CardPartForm AFTER))
             ,@Body)])

(DEFMACRO NC.ReadPtr (Stream NumBytes)
   "Reads NumBytes worth of bytes from Stream"
   [COND
      [(EQ NumBytes 1)
       `(BIN ,Stream]
      [(<= 2 NumBytes 3)
       `(CL:LOGIOR ,@[for i from (SUB1 NumBytes) to 1 by -1 collect `(LLSH (BIN ,Stream)
                                                                          ,(TIMES i 8]
                (BIN ,Stream]
      (T `(PLUS ,@[for i from (SUB1 NumBytes) to 1 by -1 collect `(LSH (BIN ,Stream)
                                                                      ,(TIMES i 8]
                 (BIN ,Stream])

(DEFMACRO NC.WritePtr (Stream Ptr NumBytes)

;;; Write down to the stream the bottom NumBytes worth of Ptr.

   [PROGN (OR NumBytes (SETQ NumBytes 3))
          (COND
             [(EQ NumBytes 1)
              `(BOUT ,Stream (LOGAND 255 ,Ptr]
             [(<= 2 NumBytes 3)
              `(PROGN ,@[for i from (SUB1 NumBytes) to 1 by -1
                            collect `(BOUT ,Stream (LOGAND 255 (LRSH ,Ptr ,(TIMES i 8]
                      (BOUT ,Stream (LOGAND 255 ,Ptr]
             (T `(PROGN ,@[for i from (SUB1 NumBytes) to 1 by -1
                              collect `(BOUT ,Stream (LOGAND 255 (RSH ,Ptr ,(TIMES i 8]
                        (BOUT ,Stream (LOGAND 255 ,Ptr])

(DEFMACRO NC.ReadStatus (Stream)

;;; Read 1 byte from Stream and return the corresponding status atom.

   `(SELCHARQ (BIN ,Stream)
         (A 'ACTIVE)
         (F 'FREE)
         (D 'DELETED)
         (S 'SPECIAL)
         NIL))

;; The following aren't called anymore, but are saved for reference purposes.

(DECLARE%: EVAL@COMPILE

(PUTPROPS NC.GetPtr MACRO [X (CONS 'IPLUS (for I from (COND
                                                        ((CADR X)
                                                         (SUB1 (CADR X)))
                                                        (T 2))
                                          to 0 by -1 collect (COND
                                                               ((ZEROP I)
                                                                (LIST 'BIN (CAR X)))
                                                               (T (LIST 'LLSH (LIST 'BIN (CAR X))
                                                                        (ITIMES 8 I])

(PUTPROPS NC.PutPtr MACRO [X (CONS 'PROGN (for I from (COND
                                                        ((CADDR X)
                                                         (SUB1 (CADDR X)))
                                                        (T 2))
                                          to 0 by -1
                                          collect (LIST 'BOUT (CAR X)
                                                        (LIST 'LOGAND 255 (COND
                                                                            ((ZEROP I)
                                                                             (CADR X))
                                                                            (T (LIST 'RSH (CADR X)
                                                                                     (ITIMES 8 I])

(PUTPROPS NC.GetStatus MACRO (X (LIST 'SELCHARQ (LIST 'BIN (CAR X))
                                      '(A 'ACTIVE)
                                      '(F 'FREE)
                                      '(D 'DELETED)
                                      '(S 'SPECIAL)
```

```
                                    'NIL)))

(PUTPROPS NC.PutStatus MACRO [X (LIST 'BOUT (CAR X)
                                    (SELECTQ (CADR X)
                                         ((A ACTIVE)
                                              (CONSTANT (CHARCODE A)))
                                         ((D DELETED)
                                              (CONSTANT (CHARCODE D)))
                                         ((F FREE)
                                              (CONSTANT (CHARCODE F)))
                                         ((S SPECIAL)
                                              (CONSTANT (CHARCODE S)))
                                         (NILL])
)
```

;; used in NCCOMPACT

```
(DECLARE%: EVAL@COMPILE

(RECORD SortingRecord (FileLoc Card CardPartTypeNum))
)
```

;; used in NCREPAIR

```
(DECLARE%: EVAL@COMPILE

(DATATYPE ScavengerInfo (MainDataInfo TitleInfo LinksInfo PropListInfo))

(RECORD LINKSDATA (CARD VERSIONDATE TOLINKS FROMLINKS GLOBALLINKS))

(RECORD TITLEDATA (CARD VERSIONDATE TITLE))

(RECORD PROPLISTDATA (CARD VERSIONDATE PROPLIST))
)

(/DECLAREDATATYPE 'ScavengerInfo '(POINTER POINTER POINTER POINTER)
        ;; ---field descriptor list elided by lister---
        '8)
```

;; used in NCCARDS

```
(DECLARE%: EVAL@COMPILE

(DATATYPE CardObject (UID NoteFile Monitor Status (IndexDirtyFlg FLAG)
                          (IndexLoc INTEGER)
                          (MainLoc INTEGER)
                          (LinksLoc INTEGER)
                          (PropListLoc INTEGER)
                          (TitleLoc INTEGER)
                          Title
                          (TitleDirtyFlg FLAG)
                          Type
                          (ActiveFlg FLAG)
                          CardCache UserData))

(DATATYPE CardCache (Substance Links PropList Region SavedRegion (ItemDate FIXP)
                           (LinksDate FIXP)
                           (TitleDate FIXP)
                           (PropListDate FIXP)
                           (ActiveCardFlg FLAG)
                           (LinksDirtyFlg FLAG)
                           (PropListDirtyFlg FLAG)
                           (SubstanceDirtyFlg FLAG)
                           (NewCardFlg FLAG)
                           (BeingDeletedFlg FLAG)))

(ACCESSFNS Card [

          (* * Instance variables of the card object)

                 (UID (fetch (CardObject UID) of DATUM)
                      (replace (CardObject UID) of DATUM with NEWVALUE))
                 (NoteFile [LET ((\NF (fetch (CardObject NoteFile) of DATUM)))
                                (COND
                                   ((type? NoteFile \NF)
                                    \NF)
                                   (\NF (replace (CardObject NoteFile) of DATUM with (NC.NoteFileFromNoteFileUID
                                                                                         \NF]
                        (replace (CardObject NoteFile) of DATUM with NEWVALUE))
                 (Monitor (fetch (CardObject Monitor) of DATUM)
                        (replace (CardObject Monitor) of DATUM with NEWVALUE))
                 (Status (fetch (CardObject Status) of DATUM)
                        (replace (CardObject Status) of DATUM with NEWVALUE))
                 (IndexDirtyFlg (fetch (CardObject IndexDirtyFlg) of DATUM)
                        (replace (CardObject IndexDirtyFlg) of DATUM with NEWVALUE))
```

```
(IndexLoc (fetch (CardObject IndexLoc) of DATUM)
       (replace (CardObject IndexLoc) of DATUM with NEWVALUE))
(MainLoc (fetch (CardObject MainLoc) of DATUM)
       (replace (CardObject MainLoc) of DATUM with NEWVALUE))
(LinksLoc (fetch (CardObject LinksLoc) of DATUM)
       (replace (CardObject LinksLoc) of DATUM with NEWVALUE))
(PropListLoc (fetch (CardObject PropListLoc) of DATUM)
       (replace (CardObject PropListLoc) of DATUM with NEWVALUE))
(TitleLoc (fetch (CardObject TitleLoc) of DATUM)
       (replace (CardObject TitleLoc) of DATUM with NEWVALUE))
(Title (fetch (CardObject Title) of DATUM)
       (replace (CardObject Title) of DATUM with NEWVALUE))
(TitleDirtyFlg (fetch (CardObject TitleDirtyFlg) of DATUM)
       (replace (CardObject TitleDirtyFlg) of DATUM with NEWVALUE))
```

(* Note the use of NC.GetType if CardType is NIL. This is required to force the access to go to the disk for the card type if the card type caching process is not finished before this fetch is being made.
Sort of a kludge.)

```
(Type (OR (fetch (CardObject Type) of DATUM)
          (NC.GetType DATUM))
      (replace (CardObject Type) of DATUM with NEWVALUE))
(ActiveFlg (fetch (CardObject ActiveFlg) of DATUM)
       (replace (CardObject ActiveFlg) of DATUM with NEWVALUE))
(ActiveCardFlg (fetch (CardObject ActiveFlg) of DATUM)
       (replace (CardObject ActiveFlg) of DATUM with NEWVALUE))
(CardCache (fetch (CardObject CardCache) of DATUM)
       (replace (CardObject CardCache) of DATUM with NEWVALUE))
(UserData (fetch (CardObject UserData) of DATUM)
       (replace (CardObject UserData) of DATUM with NEWVALUE))
[Substance (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                (AND \Cache (fetch (CardCache Substance) of \Cache)))
       (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
            (COND
               (\Cache (replace (CardCache Substance) of \Cache with NEWVALUE))
               (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                        Substance _ NEWVALUE))
                  NEWVALUE]
[Links (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
            (AND \Cache (fetch (CardCache Links) of \Cache)))
       (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
            (COND
               (\Cache (replace (CardCache Links) of \Cache with NEWVALUE))
               (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                        Links _ NEWVALUE))
                  NEWVALUE]
[FromLinks (LET ((\Cache (fetch (CardObject CardCache) of DATUM))
                 \Links)
                (AND \Cache (SETQ \Links (fetch (CardCache Links) of \Cache))
                     (fetch (LinksCache FromLinks) of \Links)))
       (LET ([\Cache (OR (fetch (CardObject CardCache) of DATUM)
                         (replace (CardObject CardCache) of DATUM with (create CardCache]
             \Links)
            (SETQ \Links (fetch (CardCache Links) of \Cache))
            (COND
               (\Links (replace (LinksCache FromLinks) of \Links with NEWVALUE))
               (T (replace (CardCache Links) of \Cache with (create LinksCache
                                                                    FromLinks _ NEWVALUE))
                  NEWVALUE]
[ToLinks (LET ((\Cache (fetch (CardObject CardCache) of DATUM))
               \Links)
              (AND \Cache (SETQ \Links (fetch (CardCache Links) of \Cache))
                   (fetch (LinksCache ToLinks) of \Links)))
       (LET ([\Cache (OR (fetch (CardObject CardCache) of DATUM)
                         (replace (CardObject CardCache) of DATUM with (create CardCache]
             \Links)
            (SETQ \Links (fetch (CardCache Links) of \Cache))
            (COND
               (\Links (replace (LinksCache ToLinks) of \Links with NEWVALUE))
               (T (replace (CardCache Links) of \Cache with (create LinksCache
                                                                    ToLinks _ NEWVALUE))
                  NEWVALUE]
[GlobalLinks (LET ((\Cache (fetch (CardObject CardCache) of DATUM))
                   \Links)
                  (AND \Cache (SETQ \Links (fetch (CardCache Links) of \Cache))
                       (fetch (LinksCache GlobalLinks) of \Links)))
       (LET ([\Cache (OR (fetch (CardObject CardCache) of DATUM)
                         (replace (CardObject CardCache) of DATUM with (create CardCache]
             \Links)
            (SETQ \Links (fetch (CardCache Links) of \Cache))
            (COND
               (\Links (replace (LinksCache GlobalLinks) of \Links with NEWVALUE))
               (T (replace (CardCache Links) of \Cache with (create LinksCache
                                                                    GlobalLinks _ NEWVALUE))
                  NEWVALUE]
[PropList (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
               (AND \Cache (fetch (CardCache PropList) of \Cache)))
```

```
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache PropList) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     PropList _ NEWVALUE))
                               NEWVALUE]
              [Region (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                           (AND \Cache (fetch (CardCache Region) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache Region) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     Region _ NEWVALUE))
                               NEWVALUE]
              [SavedRegion (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                                (AND \Cache (fetch (CardCache SavedRegion) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache SavedRegion) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     SavedRegion _ NEWVALUE))
                               NEWVALUE]
              [ItemDate (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                             (AND \Cache (fetch (CardCache ItemDate) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache ItemDate) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     ItemDate _ NEWVALUE))
                               NEWVALUE]
              [LinksDate (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                              (AND \Cache (fetch (CardCache LinksDate) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache LinksDate) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     LinksDate _ NEWVALUE))
                               NEWVALUE]
              [TitleDate (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                              (AND \Cache (fetch (CardCache TitleDate) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache TitleDate) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     TitleDate _ NEWVALUE))
                               NEWVALUE]
              [PropListDate (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                                 (AND \Cache (fetch (CardCache PropListDate) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache PropListDate) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     PropListDate _ NEWVALUE))
                               NEWVALUE]
              [LinksDirtyFlg (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                                  (AND \Cache (fetch (CardCache LinksDirtyFlg) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache LinksDirtyFlg) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     LinksDirtyFlg _ NEWVALUE))
                               NEWVALUE]
              [PropListDirtyFlg (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                                     (AND \Cache (fetch (CardCache PropListDirtyFlg) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache PropListDirtyFlg) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     PropListDirtyFlg _
                                                                                     NEWVALUE))
                               NEWVALUE]
              [SubstanceDirtyFlg (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                                      (AND \Cache (fetch (CardCache SubstanceDirtyFlg) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache SubstanceDirtyFlg) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     SubstanceDirtyFlg _
                                                                                     NEWVALUE))
                               NEWVALUE]
              [NewCardFlg (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                               (AND \Cache (fetch (CardCache NewCardFlg) of \Cache)))
                    (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                         (COND
                            (\Cache (replace (CardCache NewCardFlg) of \Cache with NEWVALUE))
                            (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                     NewCardFlg _ NEWVALUE))
```

```
                                NEWVALUE]
                    [BeingDeletedFlg (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                                          (AND \Cache (fetch (CardCache BeingDeletedFlg) of \Cache)))
                           (LET ((\Cache (fetch (CardObject CardCache) of DATUM)))
                                (COND
                                   (\Cache (replace (CardCache BeingDeletedFlg) of \Cache with NEWVALUE))
                                   (T (replace (CardObject CardCache) of DATUM with (create CardCache
                                                                                            BeingDeletedFlg _ NEWVALUE
                                                                                            ))
                                      NEWVALUE]

        (* * Class variables of the card object {class == card type})

                    (SuperType (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                    (NC.GetCardTypeField SuperType NoteCardType)))
                    (LinkDisplayMode (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                          (NC.GetCardTypeField LinkDisplayMode NoteCardType)))
                    (DefaultWidth (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                       (NC.GetCardTypeField DefaultWidth NoteCardType)))
                    (DefaultHeight (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                        (NC.GetCardTypeField DefaultHeight NoteCardType)))
                    (LinkAnchorModesSupported (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                                   (NC.GetCardTypeField LinkAnchorModesSupported NoteCardType)))
                    (LinkIconAttachedBitMap (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                                 (NC.GetCardTypeField LinkIconAttachedBitMap NoteCardType)))
                    (LeftButtonMenuItems (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                              (NC.GetCardTypeField LeftButtonMenuItems NoteCardType)))
                    (MiddleButtonMenuItems (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                                (NC.GetCardTypeField MiddleButtonMenuItems NoteCardType)))

        (* * Methods of the card object {class == card type})

                    (MakeFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                 (NC.GetCardTypeField MakeFn NoteCardType)))
                    (EditFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                 (NC.GetCardTypeField EditFn NoteCardType)))
                    (QuitFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                 (NC.GetCardTypeField QuitFn NoteCardType)))
                    (MakeReadOnlyFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                         (NC.GetCardTypeField MakeReadOnlyFn NoteCardType)))
                    (MakeReadWriteFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                          (NC.GetCardTypeField MakeReadWriteFn NoteCardType)))
                    (GetFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                (NC.GetCardTypeField GetFn NoteCardType)))
                    (PutFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                (NC.GetCardTypeField PutFn NoteCardType)))
                    (CopyFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                 (NC.GetCardTypeField CopyFn NoteCardType)))
                    (MarkDirtyFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                      (NC.GetCardTypeField MarkDirtyFn NoteCardType)))
                    (DirtyPFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                   (NC.GetCardTypeField DirtyPFn NoteCardType)))
                    (CollectLinksFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                         (NC.GetCardTypeField CollectLinksFn NoteCardType)))
                    (DeleteLinksFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                        (NC.GetCardTypeField DeleteLinksFn NoteCardType)))
                    (UpdateLinkIconsFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                            (NC.GetCardTypeField UpdateLinkIconsFn NoteCardType)))
                    (InsertLinkFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                       (NC.GetCardTypeField InsertLinkFn NoteCardType)))
                    (TranslateWindowPositionFn (LET ((NoteCardType (fetch (Card Type) of DATUM)))
                                                     (NC.GetCardTypeField TranslateWindowPositionFn NoteCardType)))

        (* * Methods inherited from the cards current NoteFile device)

                    (MarkCardDeletedFn (fetch (NoteFile MarkCardDeletedFn) of (fetch (Card NoteFile) of DATUM)))
                    (GetCardInfoFn (fetch (NoteFile GetCardInfoFn) of (fetch (Card NoteFile) of DATUM)))
                    (PutCardPartFn (fetch (NoteFile PutCardPartFn) of (fetch (Card NoteFile) of DATUM)))
                    (GetCardPartFn (fetch (NoteFile GetCardPartFn) of (fetch (Card NoteFile) of DATUM)))
                    (ObtainWritePermissionFn (fetch (NoteFile ObtainWritePermissionFn) of (fetch (Card NoteFile)
                                                                                                 of DATUM)))
                    (ReleaseWritePermissionFn (fetch (NoteFile ReleaseWritePermissionFn) of (fetch (Card NoteFile)
                                                                                                   of DATUM)))
                    (CancelCacheSubscriptionFn (fetch (NoteFile CancelCacheSubscriptionFn)
                                                      of (fetch (Card NoteFile) of DATUM]
        (CREATE (create CardObject))
        (TYPE? (TYPE? CardObject DATUM)))

(DATATYPE LinksCache (ToLinks FromLinks GlobalLinks))
)

(/DECLAREDATATYPE 'CardObject
        '(POINTER POINTER POINTER POINTER FLAG FIXP FIXP FIXP FIXP FIXP POINTER FLAG POINTER FLAG POINTER POINTER
                )
        ;; ---field descriptor list elided by lister---
        '26)
```

```
(/DECLAREDATATYPE 'CardCache '(POINTER POINTER POINTER POINTER POINTER FIXP FIXP FIXP FIXP FLAG FLAG FLAG FLAG
                                  FLAG FLAG)
        ;; ---field descriptor list elided by lister---
        '18)

(/DECLAREDATATYPE 'LinksCache '(POINTER POINTER POINTER)
        ;; ---field descriptor list elided by lister---
        '6)
```

(DEFMACRO **NC.WithWritePermission** (CardForm CardPartForm &REST Body)

;;; Evaluate Body after obtaining WritePermission for the CardPart of a Card. Release Write permission afterwards.

;; fgh 8/30/86 First created.

```
`(RESETLST
     [RESETSAVE (NC.ApplyFn ObtainWritePermissionFn ,CardForm ,CardPartForm)
            `(APPLY* ,(fetch (Card ReleaseWritePermissionFn) of ,CardForm)
                   ,,CardForm
                   ,',CardPartForm]
     ,@Body))
```

(DEFMACRO **NC.IfCardPartNotBusy** (CardForm CardPartForm &REST Body)

;; Do Body if can obtain write permission for CardPart of Card. Otherwise call CardPartBusy.

;; fgh 8/30/86 First created.

```
`[COND
    ((NOT (NC.ApplyFn ObtainWritePermissionFn ,CardForm ,CardPartForm))
     (NC.CardPartBusy ,CardForm ,CardPartForm)
     NIL)
    (T (RESETLST
           [RESETSAVE NIL `(APPLY* ,(fetch (Card ReleaseWritePermissionFn) of ,CardForm)
                                  ,,CardForm
                                  ,,CardPartForm]
           ,@Body)])
```

(DEFMACRO **NC.IfMultipleCardPartsNotBusy** (CardForm CardPartsList &REST Body)

;; Essentially call IfCardPartNotBusy for a whole bunch of CardParts

;; fgh 8/30/86 First created.

```
(LET [(FormUnderConstruction `(PROGN ,@Body]
     [for CardPart in (REVERSE CardPartsList) do (SETQ FormUnderConstruction `(NC.IfCardPartNotBusy
                                                                               ,CardForm
                                                                               ',CardPart
                                                                               ,FormUnderConstruction]

     FormUnderConstruction))
```

;; Hash Array Handler

(DEFMACRO **NC.MapCards** (NoteFile Function &OPTIONAL CollectResultPredicate)

;; Map over the entries in a NoteFiles hash array applying function to the CardID and Hash table value for each entry.

;; rht 7/14/86: Changed from function to macro.

;; fgh 7/16/86 Fixed several bugs and reorganized macro.

```
[COND
   (CollectResultPredicate `(LET (CollectionList)
                                 [MAPHASH (fetch (NoteFile HashArray) of ,NoteFile)
                                        (FUNCTION (LAMBDA (Value Key)
                                                          (LET (PredicateResult)
                                                               (BLOCK)
                                                               (COND
                                                                  ((SETQ PredicateResult (APPLY*
                                                                                           ,
                                                                                         CollectResultPredicate
                                                                                           Value))
                                                                   (SETQ CollectionList (CONS (APPLY* ,Function
                                                                                                      Value
                                                                                                     PredicateResult
                                                                                                      )
                                                                                              CollectionList)))
                                                                  (T (APPLY* ,Function Value]
                                 CollectionList))
    (T `(MAPHASH (fetch (NoteFile HashArray) of ,NoteFile)
                (FUNCTION (LAMBDA (Value Key)
                                  (BLOCK)
                                  (APPLY* ,Function Value])
```

```
(DECLARE%: EVAL@COMPILE
```

```
(RECORD PropListItem (PropertyName Value OriginalListFlg AllowEditFlg AllowSelectFlg ButtonFn))
)
```

;; used in NCLINKS

```
(DECLARE%: EVAL@COMPILE

(DATATYPE Link (UID SourceCard DestinationCard AnchorMode Label DisplayMode UserData))

(TYPERECORD LINKDISPLAYMODE (SHOWTITLEFLG SHOWLINKTYPEFLG ATTACHBITMAPFLG)
       (TYPE? (AND (EQ (LENGTH DATUM)
                       4)
                   (EQ (CAR DATUM)
                       'LINKDISPLAYMODE)
                   (FMEMB (CADR DATUM)
                          '(T NIL FLOAT))
                   (FMEMB (CADDR DATUM)
                          '(T NIL FLOAT))
                   (FMEMB (CADDDR DATUM)
                          '(T NIL FLOAT))
                   T)))
)

(/DECLAREDATATYPE 'Link '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '14)

(DECLARE%: EVAL@COMPILE

(DATATYPE NCPointer (DestinationCard DisplayMode Label UserData))
)

(/DECLAREDATATYPE 'NCPointer '(POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '8)
```

;; used in NCTYPESMECH

```
(DECLARE%: EVAL@COMPILE

(DATATYPE NoteCardType (TypeName SuperType StubFlg FullDefinitionFile MakeFn (MakeFnInheritedFlg FLAG)
                               EditFn
                               (EditFnInheritedFlg FLAG)
                               QuitFn
                               (QuitFnInheritedFlg FLAG)
                               MakeReadOnlyFn
                               (MakeReadOnlyFnInheritedFlg FLAG)
                               MakeReadWriteFn
                               (MakeReadWriteFnInheritedFlg FLAG)
                               GetFn
                               (GetFnInheritedFlg FLAG)
                               PutFn
                               (PutFnInheritedFlg FLAG)
                               CopyFn
                               (CopyFnInheritedFlg FLAG)
                               MarkDirtyFn
                               (MarkDirtyFnInheritedFlg FLAG)
                               DirtyPFn
                               (DirtyPFnInheritedFlg FLAG)
                               CollectLinksFn
                               (CollectLinksFnInheritedFlg FLAG)
                               DeleteLinksFn
                               (DeleteLinksFnInheritedFlg FLAG)
                               UpdateLinkIconsFn
                               (UpdateLinkIconsFnInheritedFlg FLAG)
                               InsertLinkFn
                               (InsertLinkFnInheritedFlg FLAG)
                               TranslateWindowPositionFn
                               (TranslateWindowPositionFnInheritedFlg FLAG)
                               LinkDisplayMode
                               (LinkDisplayModeInheritedFlg FLAG)
                               DefaultWidth
                               (DefaultWidthInheritedFlg FLAG)
                               DefaultHeight
                               (DefaultHeightInheritedFlg FLAG)
                               LinkAnchorModesSupported
                               (LinkAnchorModesSupportedInheritedFlg FLAG)
                               DisplayedInMenuFlg
                               (DisplayedInMenuFlgInheritedFlg FLAG)
                               LinkIconAttachedBitMap
                               (LinkIconAttachedBitMapInheritedFlg FLAG)
                               LeftButtonMenuItems
                               (LeftButtonMenuItemsInheritedFlg FLAG)
                               MiddleButtonMenuItems
```

```
                              (MiddleButtonMenuItemsInheritedFlg FLAG))
       MakeFnInheritedFlg _ T EditFnInheritedFlg _ T QuitFnInheritedFlg _ T MakeReadOnlyFnInheritedFlg _ T
       MakeReadWriteFnInheritedFlg _ T GetFnInheritedFlg _ T PutFnInheritedFlg _ T CopyFnInheritedFlg _ T
       MarkDirtyFnInheritedFlg _ T DirtyPFnInheritedFlg _ T CollectLinksFnInheritedFlg _ T
       DeleteLinksFnInheritedFlg _ T UpdateLinkIconsFnInheritedFlg _ T InsertLinkFnInheritedFlg _ T
       TranslateWindowPositionFnInheritedFlg _ T LinkDisplayModeInheritedFlg _ T DefaultWidthInheritedFlg _ T
       DefaultWidthInheritedFlg _ T DefaultHeightInheritedFlg _ T LinkAnchorModesSupportedInheritedFlg _ T
       LinkIconAttachedBitMapInheritedFlg _ T LeftButtonMenuItemsInheritedFlg _ T
       MiddleButtonMenuItemsInheritedFlg _ T DisplayedInMenuFlgInheritedFlg _ NIL MakeFn _ '\\FILLME// EditFn _
       '\\FILLME// QuitFn _ '\\FILLME// MakeReadOnlyFn _ '\\FILLME// MakeReadWriteFn _ '\\FILLME// GetFn _
       '\\FILLME// PutFn _ '\\FILLME// CopyFn _ '\\FILLME// MarkDirtyFn _ '\\FILLME// DirtyPFn _ '\\FILLME//
       CollectLinksFn _ '\\FILLME// DeleteLinksFn _ '\\FILLME// UpdateLinkIconsFn _ '\\FILLME// InsertLinkFn _
       '\\FILLME// TranslateWindowPositionFn _ '\\FILLME// LinkDisplayMode _ '\\FILLME// DefaultWidth _
       '\\FILLME// DefaultHeight _ '\\FILLME// LinkAnchorModesSupported _ '\\FILLME// DisplayedInMenuFlg _
       '\\FILLME// LinkIconAttachedBitMap _ '\\FILLME// LeftButtonMenuItems _ '\\FILLME// MiddleButtonMenuItems
       _ '\\FILLME//)
)

(/DECLAREDATATYPE 'NoteCardType
       '(POINTER POINTER POINTER POINTER POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG
              POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG
              POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG
              POINTER FLAG POINTER FLAG POINTER FLAG POINTER FLAG)
       ;; ---field descriptor list elided by lister---
       '54)

(DECLARE%: EVAL@COMPILE

(PUTPROPS NC.ApplyFn MACRO [Args `(APPLY* (fetch (Card ,(CAR Args)) of ,(CADR Args))
                                          ,@(CDR Args])

(PUTPROPS NC.ApplySupersFn MACRO [Args `(LET [(SuperType (fetch (Card SuperType) of ,(CADR Args]
                                                (APPLY* (NC.GetCardTypeField ,(CAR Args)
                                                             SuperType)
                                                        ,@(CDR Args])
)


(DEFMACRO NC.GetCardTypeField (FieldName CardTypeNameForm)
   ;; Fetch the card type datatype field passed directly for FieldName (for the card type whose name is in a variable passed as the second parameter)
   ;; rht 4/11/86: Was trying to autoload if field of card type was nil. Now only tries to autoload if CardTypeRecord is nil or if StubFlg is non-nil.
   ;; fgh 4/25/86 Fix to above fix. Checks for CardTypeRecord before doing fetch's in 2 cluase of COND.
   ;; kirk&fgh 26Jun86 Added check for InheritedFlg to above fix. Changed to a DEFMACRO
   ;; fgh 8/26/86 Revamped completely to clean up and to account for case where one of supertypes is a stub and must be autoloaded. Added ability to
   ;; handle \\EMPTY// fields.
   ;; kirk 8/26/86 Added check for FMEMB of FieldName in (RECORDFIELDNAMES (QUOTE NoteCardTYpe)) before evaling fetch
   ;; rht 11/1/86: Added check for NIL CardTypeName. Also checks whether card is top level NoteCard type before fetching from super type.
   ;; rht 11/9/86: Totally revamped to use \\FILLME// field. No longer cares about value of StubFlg.
   `(LET ((CardTypeName ,CardTypeNameForm)
          CardTypeRecord FieldValue)
         (if CardTypeName
             then [if (OR (NULL (SETQ CardTypeRecord (NC.CardTypeRecord CardTypeName)))
                          (EQ (SETQ FieldValue (fetch (NoteCardType ,FieldName) of CardTypeRecord))
                              '\\EMPTY//))
                      then                                          ; either the card type record doesn't exist or its just a stub --
                                                                    ; either way it needs to be autoloaded.
                           (if [OR [NULL (NC.CardTypeLoader CardTypeName (AND CardTypeRecord (fetch (NoteCardType
                                                                                                     FullDefinitionFile
                                                                                                     )
                                                                                                of CardTypeRecord]
                                       (NULL (SETQ CardTypeRecord (NC.CardTypeRecord CardTypeName]
                               then (NC.ReportError NIL (CONCAT "Cannot find full definition of card type: "
                                                                CardTypeName)))
                           (if (EQ (SETQ FieldValue (fetch (NoteCardType ,FieldName) of CardTypeRecord))
                                   '\\EMPTY//)
                               then                                 ; still marked \\EMPTY//
                                    (NC.ReportError "NC.GetCardTypeField" (CONCAT "Field name "
                                                                                  ',FieldName " of card type "
                                                                                  CardTypeName " still \\EMPTY//
                                                                                  after autoloading."]
                       (if (EQ FieldValue '\\FILLME//)
                           then                                     ; Fetch field from super type.  Unfortunately direct recursion is
                                                                    ; not possible because this is a DEFMACRO.
                                (NC.GetCardTypeFieldOfSuper CardTypeName ',FieldName)
                         else FieldValue))))

;; used in NCPROGINT

(DEFMACRO NCP.ApplyCardTypeFn (Fn Card &REST Args)
```

```
    ;; Apply the cardtypefn Fn to Card and the rest of the args.
    `(NC.ApplyFn ,Fn ,Card ,@Args))

(DEFMACRO NCP.ApplySuperTypeFn (Fn Card &REST Args)

;;; Apply the cardtypefn Fn of Card's super type to the given args.

    `(NC.ApplySupersFn ,Fn ,Card ,@Args))

(DEFMACRO NCP.MapCards (NoteFile Function &OPTIONAL CollectResultPredicate)

;;; Map down all notecards (including fileboxes) in NoteFile performing Function to each. If CollectResultPredicate is non-nil, then collect the results of
;;; applying Function to those cards satisfying CollectResultPredicate.

;;; rht 7/13/86: Now only works on non-deleted cards.

;;; rht 7/17/86: rearranged slightly.

    `[NC.MapCards ,NoteFile ,[AND Function `(FUNCTION (LAMBDA (Card)
                                                        (AND (NCP.ValidCardP Card)
                                                             (APPLY* ,Function Card]
            ,(AND CollectResultPredicate `(FUNCTION (LAMBDA (Card)
                                                      (AND (NCP.ValidCardP Card)
                                                           (APPLY* ,CollectResultPredicate Card])

(DEFMACRO NCP.MapCardsOfType (Types NoteFile Function &OPTIONAL CollectResultPredicate)
    ;; Map down all fileboxes in the current notefile, performing Function to each. If CollectResultPredicate is non-nil, then collect the results of applying
    ;; Function to those cards satisfying CollectResultPredicate.
    ;; rht 7/12/86: Now takes arbitrary types to map over.
    ;; rht 7/17/86: rearranged slightly.
    `[NCP.MapCards ,NoteFile ,[AND Function `(FUNCTION (LAMBDA (Card)
                                                         (AND (FMEMB (NCP.CardType Card)
                                                                     (MKLIST ,Types))
                                                              (APPLY* (FUNCTION ,Function)
                                                                      Card]
            ,(AND CollectResultPredicate `(FUNCTION (LAMBDA (Card)
                                                      (AND (FMEMB (NCP.CardType Card)
                                                                  (MKLIST ,Types))
                                                           (APPLY* (FUNCTION ,CollectResultPredicate)
                                                                   Card])

(DEFMACRO NCP.WithLockedCards (&BODY Body)
    `(NAMED-RESETLST CardListResetVar ,@Body))

(DECLARE%: EVAL@COMPILE

(TYPERECORD NOTECARDDATES (SUBSTANCEDATE LINKSDATE TITLEDATE PROPLISTDATE))
)

(DEFMACRO NCP.MapLinks (NoteFile Function &OPTIONAL CollectResultPredicate)
    ;; Map down all links in the current notefile, performing LinkFn to each.
    ;; rht 11/18/85: Updated to handle new notefile and card object formats. Now allows collection of links satisfying CollectResultPredicate.
    ;; rht 7/17/86: rearranged slightly.
    ;; rht 1/26/87: Now uses MAPCONC rather than APPLY'ing NCONC.
    `[MAPCONC [NCP.MapCards ,NoteFile [FUNCTION (LAMBDA (Card)
                                                  (if ,CollectResultPredicate
                                                      then (for Link in (NC.RetrieveToLinks Card)
                                                              when (APPLY* ,CollectResultPredicate Link)
                                                              collect (APPLY* ,Function Link))
                                                    else (for Link in (NC.RetrieveToLinks Card)
                                                            do (APPLY* ,Function Link]
                ,(AND CollectResultPredicate `(FUNCTION TRUE]
          (FUNCTION (LAMBDA (X)
                      X]))

(DEFMACRO NCP.MapLinksOfType (Types NoteFile Function &OPTIONAL CollectResultPredicate)
    ;; Map down all links in the given notefile, having one of the given types.
    ;; rht 7/17/86: rearranged slightly.
    ;; rht 1/26/87: Now uses MAPCONC rather than APPLY'ing NCONC.
    `[MAPCONC [NCP.MapCards ,NoteFile [FUNCTION (LAMBDA (Card)
                                                  (if ,CollectResultPredicate
                                                      then (for Link in (NC.RetrieveToLinks Card)
                                                              when (AND (FMEMB (NCP.LinkType Link)
                                                                               (MKLIST ,Types))
```

```
                                                                    (APPLY* ,CollectResultPredicate Link))
                                                            collect (APPLY* ,Function Link))
                                                    else (for Link in (NC.RetrieveToLinks Card)
                                                            when (FMEMB (NCP.LinkType Link)
                                                                        (MKLIST ,Types))
                                                            do (APPLY* ,Function Link]
                    ,(AND CollectResultPredicate `(FUNCTION TRUE]
            (FUNCTION (LAMBDA (X)
                        X])
```

;; used in NCCROSSFILELINKS

```
(DECLARE%: EVAL@COMPILE

(DATATYPE CrossFileLinkSubstance (CrossFileLinkDestCardUID CrossFileLinkDestNoteFileUID
                                        CrossFileLinkDestFileHint (CrossFileLinkTwoWayFlg FLAG)
                                        RemoteCrossFileLinkCardUID))
)

(/DECLAREDATATYPE 'CrossFileLinkSubstance '(POINTER POINTER POINTER FLAG POINTER)
       ;; ---field descriptor list elided by lister---
       '8)
```

;; used in NCBROWSERCARD

```
(DECLARE%: EVAL@COMPILE

(RECORD SPECIALBROWSERSPECS (Font MotherD PersonalD FamilyD)
       PersonalD _ 10)
)
```

;; used in NCCONVERTVERSION2TO3

```
(DECLARE%: EVAL@COMPILE

(RECORD POINTERLIST (STATUS MAINPTR LINKSPTR TITLEPTR PROPSPTR INDEXPTR))

[TYPERECORD NOTECARDLINK (LINKID SOURCEID DESTINATIONID ANCHORMODE LINKLABEL DISPLAYMODE)
       (TYPE? (AND (EQ (LENGTH DATUM)
                       7)
                   (NC.IDP (fetch (NOTECARDLINK SOURCEID) of DATUM))
                   (NC.IDP (fetch (NOTECARDLINK DESTINATIONID) of DATUM]
)

(PUTPROPS NCDECLS FILETYPE :FAKE-COMPILE-FILE)

(PUTPROPS NCDECLS MAKEFILE-ENVIRONMENT (:PACKAGE "IL" :READTABLE "INTERLISP" :BASE 10))

(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA LOADINITADVISE LOADINITPROPS)

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

(PUTPROPS NCDECLS COPYRIGHT ("Venue & Xerox Corporation" 1987 1988 1989 1990 1993 1994 2020))
```

FUNCTION INDEX

MACRO INDEX

RECORD INDEX

PROPERTY INDEX