File created: 5-Nov-2020 19:58:43 **{DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCBROWSERCARD.;6**

previous date: 9-Jan-94 20:07:43 {DSK}<users>arunwelch>skydrive>documents>unix>lisp>lde>notecards>system>NCBROWSERCARD.;5

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

;;


```
(RPAQQ NCBROWSERCARDCOMS
       (
```

;;; Stuff for the Notecards browser.

```
        [DECLARE%: DONTEVAL@LOAD FIRST (P (NC.LoadFileFromDirectories 'NCGRAPHCARD]
        (FNS NCAddStub.BrowserCard)
        (GLOBALVARS NC.SelectingBrowserSourceMenu NC.SpecialBrowserSpecsFlg NC.BrowserContentsLinkLabel
               NC.SubBoxLinkLabel NC.FiledCardLinkLabel NC.SelectingCardMenu NC.DashingStyles
               NC.GraphFlowerLinkSeparation NC.LinkDashingInBrowser NC.ArrowHeadsInBrowser
               NC.BrowserFormatOptions NC.*Graph*BrowserFormat NC.GraphEditMenuItems
               NC.GraphEditUnfixedMenuItems NC.LinkIconShowTitleFlg NC.LinkIconShowLinkTypeFlg
               NC.LinkIconAttachBitmapFlg NC.BrowserHashArraySize NC.UseDeletedLinkIconIndicatorsFlg)
        [INITVARS (NC.LinkDashingInBrowser NIL)
               (NC.ArrowHeadsInBrowser 'None)
               (NC.BrowserHashArraySize 2000)
               [NC.DashingStyles '(NIL (4 4)
                                       (10 4 2 4)
                                       (1 4)
                                       (4 10)
                                       (8 4)
                                       (14 4)
                                       (10 4 2 4 2 4)
                                       (10 4 2 4 2 4 2 4]
               (NC.GraphFlowerLinkSeparation 8)
               [NC.BrowserFormatOptions '((*GRAPH* *GRAPH* "Build a directed graph (no virtual nodes).")
                                          (LATTICE LATTICE "Build an acyclic directed graph (uses virtual nodes
                                                 in case of cycles).")
                                          (COMPACT COMPACT "Build a forest using minimal screen space.")
                                          (FAST FAST "Build a forest, sacrificing screen space for speed.")]
               (NC.*Graph*BrowserFormat '*GRAPH*)
               [NC.GraphEditMenuItems (NC.ExpandBars '(("Create Card" NC.BrowserCreateCardFn "Create a new card
                                                               and a corresponding browser node.")
                                                       ("Insert Card" NC.BrowserAddNodeFn "Add an existing card
                                                               and its corresponding browser node.")
                                                       ("Delete Card" NC.BrowserDeleteCardFn "Delete an existing
                                                               card and its browser node.")
                                                       ("----------------" NIL "")
                                                       ("Create Link" NC.BrowserCreateLinkFn "Create a new local
                                                               link and display it in the graph."
                                                               (SUBITEMS ("Create Local Link"
                                                                                NC.BrowserCreateLinkFn "Create a
                                                                                new local link and display it in
                                                                                the graph.")
                                                                         ("Create Global Link"
                                                                                NC.BrowserCreateGlobalLinkFn
                                                                                "Create a new global link and
                                                                                display it in the graph.")))
                                                       ("Delete Link" NC.BrowserDeleteLinkFn "Delete an existing
                                                               link and erase it from the graph.")
                                                       ("----------------" NIL "")
                                                       ("Move Node" NC.BrowserMoveNodeFn "Move a browser node."
                                                               (SUBITEMS ("Move Single Node" NC.BrowserMoveNodeFn
                                                                                "Move a browser node.")
                                                                         ("Move Node & SubTree"
                                                                                NC.BrowserMoveSubtreeFn "Move a
                                                                                subtree of nodes following the
                                                                                movement of the root.")
                                                                         ("Move Region"
                                                                                NC.BrowserMoveNodesInRegionFn
                                                                                "Move all nodes within a specified
                                                                                region to another region.")))
                                                       ("Remove Node" NC.BrowserRemoveNodeFn "Remove a browser
                                                               node (no card deleted).")
                                                       ("Connect Nodes" NC.BrowserAddEdgeFn "Connect two browser
                                                               nodes (no link created).")
                                                       ("Disconnect Nodes" NC.BrowserRemoveEdgeFn "Disconnect
                                                               two browser nodes (no link deleted).")
                                                       ("----------------" NIL "")
                                                       ("Add Label" NC.BrowserAddLabelFn "Add a string label to
                                                               graph.")
```

```
                                                ("Change Label" NC.BrowserChangeLabelFn "Change a
                                                      labe..")
                                                ("Smaller Label" NC.BrowserShrinkLabelFn "Reduce font
                                                      size for a browser label.")
                                                ("Larger Label" NC.BrowserGrowLabelFn "Increase font size
                                                      for a browser label.")
                                                ("Toggle Shade" NC.BrowserToggleShadeFn "Inverts label
                                                      shade for a browser label."]
            (NC.GraphEditUnfixedMenuItems (NC.ExpandBars (APPEND NC.GraphEditMenuItems
                                                                 '(BAR (FIX% MENU NC.BrowserFixGraphEditMenuFn
                                                                              "Attach this menu to edge of
                                                                              browser window."]
```

;;; BROWSER mechanisms

```
        (FNS NC.MakeBrowserCard NC.BringUpBrowserCard)
        (FNS NC.GrowLinkLattice NC.UpdateBrowserCard NC.RelayoutBrowserCard NC.LayoutNewBrowserNodes
             NC.ConnectNodesInBrowser NC.UnconnectNodesInBrowser NC.ExpandBrowserNode NC.AskBrowserSpecs
             NC.ChangeBrowserSpecs NC.AskSpecialBrowserSpecs NC.BrowserFlipRoots NC.ChangeBrowserRoots
             NC.RespecifyBrowserRoots NC.RebuildFromNodesInGraph NC.RemoveDuplicateNodesFromGraph
             NC.ShowBrowserGraph)
```

;;; Graph editor menu functions.

;;; These moved to GRAPHCARD since many are used by both.

```
        (* FNS NC.SetUpGraphEditMenus NC.GetGraphEditMenu NC.BrowserRightButtonFn NC.BrowserCreateCardFn
           NC.BrowserAddLabelFn NC.BrowserChangeLabelFn NC.BrowserAddNodeFn NC.BrowserCreateLinkFn
           NC.BrowserCreateGlobalLinkFn NC.BrowserAddLink NC.BrowserAddGlobalLink NC.BrowserAddEdgeFn
           NC.BrowserDeleteCardFn NC.BrowserRemoveNodeFn NC.BrowserDeleteLinkFn NC.BrowserRemoveEdgeFn
           NC.BrowserShrinkLabelFn NC.BrowserGrowLabelFn NC.BrowserMoveNodeFn NC.BrowserToggleShadeFn
           NC.CursorInsideGraphNodeP NC.BrowserMoveNodesInRegionFn NC.BrowserMoveSubtreeFn
           NC.BrowserFixGraphEditMenuFn NC.BrowserCreateCard NC.BrowserCreateLink NC.BrowserDeleteLink
           NC.BrowserAddNode NC.BrowserAddLabel NC.BrowserAddEdge NC.BrowserRemoveNode NC.DelBrowserContentsLink
           NC.BrowserRemoveEdge NC.BrowserChangeLabel NC.BrowserGetConfirmation)
```

;;; Grapher hacks for browser

```
        (FNS NC.MakeLinksLegend NC.MakeLinksLegendMenu NC.LinksLegendRepaintFn NC.BrowserDrawLinkFn
             NC.DrawFlowerLinks NC.DrawFlowerLink NC.LinksLegendReshapeFn NC.DrawArrowHead)
```

;;; for making and manipulating the tiny attached shrunken browser window.

```
        (INITVARS (NC.BrowserOverviewDefaultWidth 75)
               (NC.BrowserOverviewDefaultHeight 75)
               (NC.LeastScaleForGraphNodeShrinking 0.3))
        (GLOBALVARS NC.BrowserOverviewDefaultWidth NC.BrowserOverviewDefaultHeight
               NC.LeastScaleForGraphNodeShrinking NC.BrowserOverviewSpecsStylesheet
               NC.DefaultWhereToAttachOverviewWin NC.DefaultBrowserOverviewMode NC.OverviewWinMode.Compress
               NC.OverviewWinMode.Expand)
        [INITVARS (NC.OverviewWinMode.Compress '|Compress Overview Win|)
               (NC.OverviewWinMode.Expand 'Expand% Overview)
               (NC.DefaultBrowserOverviewMode 'Neither)
               (NC.DefaultWhereToAttachOverviewWin '(LEFT . TOP))
               (NC.BrowserOverviewSpecsStylesheet (CREATE.STYLE 'TITLE "Choose browser overview specs"
                                                         'ITEM.TITLES
                                                         '(Edge |Position on Edge| Mode)
                                                         'ITEM.TITLE.FONT
                                                         (FONTCOPY MENUFONT 'WEIGHT 'BOLD)
                                                         'ITEMS
                                                         (LIST [create MENU ITEMS _ '((LEFT LEFT "Position along
                                                                                             left edge.")
                                                                                      (TOP TOP "Position along
                                                                                           top edge.")
                                                                                      (BOTTOM BOTTOM
                                                                                             "Position along
                                                                                             bottom edge."]
                                                               [create MENU ITEMS _ '((TOP/RIGHT TOP
                                                                                             "Position at top or
                                                                                             right end of
                                                                                             edge.")
                                                                                      (CENTER CENTER
                                                                                             "Position at center
                                                                                             of edge.")
                                                                                      (BOTTOM/LEFT BOTTOM
                                                                                             "Position at bottom
                                                                                             or left of edge."]
                                                               (create MENU ITEMS _ '((|Compress Overview Win|
                                                                                      |Compress Overview Win|
                                                                                      "Compress the overview
                                                                                      window to exactly fit the
```

```
                                                                                                    overview contents.")
                                                                                                   (Expand% Overview
                                                                                                          Expand% Overview
                                                                                                          "Expand the
                                                                                                          overview contents
                                                                                                          to exactly fill the
                                                                                                          overview window.")
                                                                                                   (Neither Neither
                                                                                                          "Neither expand the
                                                                                                          overview contents
                                                                                                          nor compress the
                                                                                                          overview window."]
        (FNS NC.MakeBrowserOverviewWin NC.AskBrowserOverviewSpecs)
        (FNS NC.DRAWBOX NC.ShrinkGraphToWindow NC.ScaleGraphNode NC.ComputeOverviewScale
             NC.RedrawBrowserOverviewWin NC.DrawWireFrameInOverviewWin NC.CompressOverviewWin
             NC.ReattachBrowserOverviewWin NC.BrowserScrollFn NC.BrowserReshapeFn NC.BrowserOverviewWinRepaintFn
             NC.BrowserOverviewWinReshapeFn NC.BrowserOverviewWinMINSIZEFn NC.BrowserOverviewWinButtonEventFn
             NC.BrowserCardQuitFn NC.MakeBrowserCardReadOnly NC.MakeBrowserCardReadWrite)
```

;;; Miscellaneous

```
        (FNS NC.DelReferencesToCardFromBrowser NC.NewBrowserNodeUIDFromOldUID NC.GetBrowserSubstance
             NC.ComputeBrowserSavedLinkingInfo NC.ComputeBrowserSavedLinkingInfoForNode)
        (FNS NC.FetchBrowserRootsInfo NC.FetchBrowserLinkLabels NC.FetchBrowserFormat
             NC.FetchSpecialBrowserSpecs NC.FetchBrowserDepth NC.FetchBrowserSavedLinkingInfo
             NC.FetchBrowserLinksLegend)
        (FNS NC.SetBrowserRootsInfo NC.SetBrowserLinkLabels NC.SetBrowserFormat NC.SetSpecialBrowserSpecs
             NC.SetBrowserDepth NC.SetBrowserSavedLinkingInfo NC.SetBrowserLinksLegend)
        (FNS NC.ReadBrowserRootsInfo NC.ReadBrowserLinkLabels NC.ReadBrowserFormat NC.ReadSpecialBrowserSpecs
             NC.ReadBrowserDepth NC.ReadBrowserSavedLinkingInfo NC.ReadBrowserSavedLinkingInfoForNode
             NC.ReadBrowserLinksLegend)
        (FNS NC.WriteBrowserRootsInfo NC.WriteBrowserLinkLabels NC.WriteBrowserFormat
             NC.WriteSpecialBrowserSpecs NC.WriteBrowserDepth NC.WriteBrowserSavedLinkingInfo
             NC.WriteBrowserSavedLinkingInfoForNode NC.WriteBrowserLinksLegend)
        (FNS NC.GraphLinkIconUpdateCheck NC.BrowserRepaintFn NC.GetBrowserNodeID NC.MakeBrowserNodeUID
             NC.GetBrowserHashArray NC.RemoveBrowserNodeHashArrayEntry NC.HashArrayFromBrowserCard
             NC.CardFromBrowserNode NC.PutBrowserSubstance NC.FetchBrowserRoots NC.SetBrowserRoots)
        (GLOBALVARS NC.ArrowHeadLength NC.ArrowHeadAngle NC.ArrowHeadXVal NC.ArrowHeadYVal)
        [INITVARS (NC.ArrowHeadLength 7)
               (NC.ArrowHeadAngle 20)
               (NC.ArrowHeadXVal (TIMES NC.ArrowHeadLength (COS NC.ArrowHeadAngle)))
               (NC.ArrowHeadYVal (TIMES NC.ArrowHeadLength (SIN NC.ArrowHeadAngle]
```

;;; init

```
        (FNS NC.AddBrowserCard)
        (BITMAPS NC.BrowserCardIcon)
        (DECLARE%: DONTEVAL@LOAD (P (NC.AddBrowserCard)))
        (PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              NCBROWSERCARD)))
```

;;; Stuff for the Notecards browser.

```
(DECLARE%: DONTEVAL@LOAD FIRST

(NC.LoadFileFromDirectories 'NCGRAPHCARD)
)

(DEFINEQ
```

(**NCAddStub.BrowserCard**

```
  [LAMBDA NIL                                                    ; Edited  6-Dec-88 12:22 by krivacic
```

;;; kirk 18Jun86 Add the Browser card stub

;;; rht 11/7/86: Now passes down a \\FILLME// field.

```
    (NC.AddCardTypeStub 'Browser 'Graph 'NCBROWSERCARD NIL `((DisplayedInMenuFlg T)
                                                             (LinkIconAttachedBitMap ,NC.BrowserCardIcon])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.SelectingBrowserSourceMenu NC.SpecialBrowserSpecsFlg NC.BrowserContentsLinkLabel
       NC.SubBoxLinkLabel NC.FiledCardLinkLabel NC.SelectingCardMenu NC.DashingStyles
       NC.GraphFlowerLinkSeparation NC.LinkDashingInBrowser NC.ArrowHeadsInBrowser NC.BrowserFormatOptions
       NC.*Graph*BrowserFormat NC.GraphEditMenuItems NC.GraphEditUnfixedMenuItems NC.LinkIconShowTitleFlg
       NC.LinkIconShowLinkTypeFlg NC.LinkIconAttachBitmapFlg NC.BrowserHashArraySize
       NC.UseDeletedLinkIconIndicatorsFlg)
)

(RPAQ? NC.LinkDashingInBrowser NIL)
```

```
(RPAQ? NC.ArrowHeadsInBrowser 'None)

(RPAQ? NC.BrowserHashArraySize 2000)

(RPAQ? NC.DashingStyles
       '(NIL (4 4)
             (10 4 2 4)
             (1 4)
             (4 10)
             (8 4)
             (14 4)
             (10 4 2 4 2 4)
             (10 4 2 4 2 4 2 4)))

(RPAQ? NC.GraphFlowerLinkSeparation 8)

(RPAQ? NC.BrowserFormatOptions '((*GRAPH* *GRAPH* "Build a directed graph (no virtual nodes).")
                                 (LATTICE LATTICE "Build an acyclic directed graph (uses virtual nodes in case of
                                       cycles).")
                                 (COMPACT COMPACT "Build a forest using minimal screen space.")
                                 (FAST FAST "Build a forest, sacrificing screen space for speed.")))

(RPAQ? NC.*Graph*BrowserFormat '*GRAPH*)

(RPAQ? NC.GraphEditMenuItems
       [NC.ExpandBars '(("Create Card" NC.BrowserCreateCardFn "Create a new card and a corresponding browser
                               node.")
                        ("Insert Card" NC.BrowserAddNodeFn "Add an existing card and its corresponding browser
                               node.")
                        ("Delete Card" NC.BrowserDeleteCardFn "Delete an existing card and its browser node.")
                        ("----------------" NIL "")
                        ("Create Link" NC.BrowserCreateLinkFn "Create a new local link and display it in the
                               graph." (SUBITEMS ("Create Local Link" NC.BrowserCreateLinkFn "Create a new local
                                                        link and display it in the graph.")
                                                 ("Create Global Link" NC.BrowserCreateGlobalLinkFn "Create a new
                                                        global link and display it in the graph.")))
                        ("Delete Link" NC.BrowserDeleteLinkFn "Delete an existing link and erase it from the
                               graph.")
                        ("----------------" NIL "")
                        ("Move Node" NC.BrowserMoveNodeFn "Move a browser node." (SUBITEMS ("Move Single Node"
                                                                                            NC.BrowserMoveNodeFn
                                                                                            "Move a browser
                                                                                            node.")
                                                                                           ("Move Node & SubTree"
                                                                                            NC.BrowserMoveSubtreeFn
                                                                                            "Move a subtree of
                                                                                            nodes following the
                                                                                            movement of the root.")
                                                                                           ("Move Region"
                                                                                        NC.BrowserMoveNodesInRegionFn
                                                                                                    "Move all nodes
                                                                                                    within a
                                                                                                    specified region
                                                                                                    to another
                                                                                                    region.")))
                        ("Remove Node" NC.BrowserRemoveNodeFn "Remove a browser node (no card deleted).")
                        ("Connect Nodes" NC.BrowserAddEdgeFn "Connect two browser nodes (no link created).")
                        ("Disconnect Nodes" NC.BrowserRemoveEdgeFn "Disconnect two browser nodes (no link
                               deleted).")
                        ("----------------" NIL "")
                        ("Add Label" NC.BrowserAddLabelFn "Add a string label to graph.")
                        ("Change Label" NC.BrowserChangeLabelFn "Change a labe..")
                        ("Smaller Label" NC.BrowserShrinkLabelFn "Reduce font size for a browser label.")
                        ("Larger Label" NC.BrowserGrowLabelFn "Increase font size for a browser label.")
                        ("Toggle Shade" NC.BrowserToggleShadeFn "Inverts label shade for a browser label."])

(RPAQ? NC.GraphEditUnfixedMenuItems [NC.ExpandBars (APPEND NC.GraphEditMenuItems '(BAR (FIX% MENU
                                                                                        NC.BrowserFixGraphEditMenuFn
                                                                                               "Attach this menu to
                                                                                               edge of browser
                                                                                               window."])


;;; BROWSER mechanisms

(DEFINEQ

(NC.MakeBrowserCard
  [LAMBDA (Card Title NoDisplayFlg ParamList InterestedWindow RegionOrPosition)
                                                                    ; Edited 29-Sep-88 13:13 by jrc

    ;; Make a browser card with id Card using root at RootID and the link following predictae specified by Predicate.  IF Root and/or ListOfLinkLabels
    ;; not specified, ask the user.

    ;; rht 8/3/84: Changed to call NC.AskLinkLabel with its ReverseLinkLabel parameter set to T.

    ;; fgh 10/2/84 Changed Link Icons to be image objects in NodeLabel of Graph Npodes rather than annotations on graph nodes.

    ;; rht 10/19/84: Fixed setting up of browser card's prop list in case NoDisplayFlg is T so we have no Window.  Now NC.MakeLinksLegend returns
    ;; the label pairs.
```

```
;; rht 11/27/84: Removed the WINDOWADDPROP call to put NC.GraphCardCloseFn on the CLOSEFN of the window.  This causes trouble.
;; NC.QuitCard will get put on by NC.MakeNoteCard and that's enough.

;; rht 1/3/85: Now puts a dummy region of the right size if the NoDisplayFlg is on.

;; rht 1/15/85: Put hooks for AddNode, AddLink, etc.  so editing graph edits underlying structure.

;; rht 2/14/85: Added VerticalFlg and made BrowserSpecs get put on browser's proplist in all cases.

;; rht 4/1/85: Now calls NC.AskBrowserSpecs with additional Don'tAskFlg in case of call from Programmer's interface.

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

;; rht 2/7/86: Now gets browser format, etc.  via fetch/set fns.

;; rht 5/6/86: Took out call to NC.SetupTitleBarMenu.

;; rht 5/8/86: Added calls to rig title bar properly.

;; rht 7/10/86: Now passes ListOfLinkLabels to NC.AskBrowserSpecs.

;; rht 9/19/86: Mod to above fix.  Now calls NC.MakeNewCardWindow a new fn that creates window and hangs Card off windowprop.

;; pmi 12/5/86: Modified message to NC.SelectNoteCards to mention SHIFT-selection.  Also added GLOBALVARS statement.

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

;; rht 12/16/86: Now checks that NC.MakeLink succeeded before creating a real link icon.  If not, then make a standin for a cross file link icon.

;; rg 3/18/87 added NC.CardSelectionOperation wrapper

;; rht 3/20/87: Removed needless call to NC.ActivateCard.

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

;; rht 5/26/87: Now handles cross-file links properly, i.e.  uses cross-file link standin in cases when GrowLinkLattice wasn't able to follow into the
;; remote notefile.

;; dwm 1/19/88 Sets ReadOnly prop based upon current NF ReadOnly status

;; pmi 2/29/88: Now asks for specs only if none are specified coming in. A browser created from a browser now prompts for everything.  Also added
;; InterestedWindow argument.

;; rht 6/9/88: Added RegionOrPosition arg and passed to NC.MakeNewCardWindow.

;; jrc 29-sep-88.  Added WINDOWPROP calls at end of function -- in the old days, NC.RelayoutBrowserCard added these props, but
;; NC.RelayoutBrowserCard now only preserves the props that are already on the window.  The WINDOWPROP calls are copied from
;; NC.BringUpBrowserCard.
(DECLARE (GLOBALVARS NC.SubBoxLinkLabel NC.BrowserContentsLinkLabel NC.SpecialBrowserSpecsFlg
                NC.*Graph*BrowserFormat NC.SelectingBrowserSourceMenu))
(NCP.WithLockedCards
 (PROG ([RootCards (MKLIST (LISTGET ParamList 'ROOTCARDS]
        (ListOfLinkLabels (LISTGET ParamList 'LINKTYPES))
        (BrowserFormat (LISTGET ParamList 'FORMAT))
        (Depth (LISTGET ParamList 'DEPTH))
        (CardType (NC.RetrieveType Card))
        (Don'tAskForRoots (FMEMB 'ROOTCARDS ParamList))
        (Don'tAskForSpecs (OR (FMEMB 'LINKTYPES ParamList)
                              (FMEMB 'FORMAT ParamList)
                              (FMEMB 'DEPTH ParamList)))
        Lattice RootNodes Window Graph SpecialBrowserSpecs BrowserSpecs DropVirtualNodesFlg)
       (NC.IfAllCardsFree (NC.LockListOfCards RootCards "Make Browser Card")
              (if (NOT NoDisplayFlg)
                  then (SETQ Window (NC.MakeNewCardWindow Card (OR Title "Untitled")
                                           NIL NIL RegionOrPosition))
                       (WINDOWADDPROP Window 'SHRINKFN (FUNCTION NC.GraphCardShrinkFn))
                       (SETQ InterestedWindow Window))
              (OR Don'tAskForRoots (SETQ RootCards (NC.SelectNoteCards NIL NIL NC.SelectingBrowserSourceMenu
                                                     InterestedWindow "Please shift-select the Cards
                                                     and/or Boxes the browser should start from." T)))
              (COND
                 ((EQ RootCards 'DON'T)
                  (NC.DeactivateCard Card)
                  (CLOSEW Window)
                  (RETURN)))
              (NC.HoldTTYProcess)
              [SETQ BrowserSpecs (NC.AskBrowserSpecs InterestedWindow Card ListOfLinkLabels Depth
                                       BrowserFormat T (if Don'tAskForSpecs
                                                           then 'DONTASK]
              (COND
                 ((NULL BrowserSpecs)
                  (NC.DeactivateCard Card)
                  (CLOSEW Window)
                  (RETURN)))
              (SETQ ListOfLinkLabels (CAR BrowserSpecs))
              (SETQ Depth (CADR BrowserSpecs))
              (SETQ BrowserFormat (CADDR BrowserSpecs))      ; If user wants *GRAPH* format, i.e.  virtual nodes eliminated,
                                                             ; then set the flag
              (if (FMEMB NC.*Graph*BrowserFormat BrowserFormat)
                  then (SETQ DropVirtualNodesFlg T))
              [SETQ SpecialBrowserSpecs (COND
                                           (NC.SpecialBrowserSpecsFlg (NC.AskSpecialBrowserSpecs
                                                                       InterestedWindow))
                                           (T (create SPECIALBROWSERSPECS]
              (OR NoDisplayFlg (NC.PrintMsg InterestedWindow T (CHARACTER 13)
                                    "Computing browser graph. Please wait. ..."))
                                                             ; Create new browser hash array
```

```
(NC.GetBrowserHashArray Card)                    ; Compute lattice breakdth-first starting from roots.
(SETQ Lattice (NC.GrowLinkLattice RootCards NIL ListOfLinkLabels Card Depth))
(SETQ RootNodes (if RootCards
                    then (for RootCard in RootCards collect (NC.GetBrowserNodeID Card RootCard))
                  else NIL))
```

;; Link destination id information stored in NodeLabel field into a LinkIcon for display

```
(for Node in Lattice bind NodeID (CrossFileLinkModePropList _ (LIST (fetch (Card NoteFile)
                                                                       of Card)
                                                                   NIL))
   eachtime (BLOCK)
   do [replace (GRAPHNODE NODELABEL) of Node
         with (LET (NewLink)
                   (if [AND (NOT (NC.CrossFileLinkCardP (fetch (GRAPHNODE NODELABEL) of Node)))
                            (SETQ NewLink (NC.MakeLink InterestedWindow
                                                       NC.BrowserContentsLinkLabel (fetch (GRAPHNODE
                                                                                          NODELABEL)
                                                                                     of Node)
                                                       Card NIL NIL NIL NIL NIL
                                                       (NC.ComputeCrossFileLinkMode
                                                        (fetch (GRAPHNODE NODELABEL) of Node)
                                                        CrossFileLinkModePropList InterestedWindow]
                       then (NC.MakeLinkIcon NewLink)
                     else (NC.MakeCrossFileLinkIconStandIn (fetch (GRAPHNODE NODELABEL)
                                                              of Node]
                                                       ; Untouch each graph node so that next Recompute will put fresh
                                                       ; values on proplist.
      (SETQ NodeID (fetch (GRAPHNODE NODEID) of Node))
      (NC.GraphNodeIDRemProp (NC.CoerceToGraphNodeID NodeID)
             'TouchedFlg)
      (NC.GraphNodeIDRemProp (NC.CoerceToGraphNodeID NodeID)
             'VisitedFlg))
(SETQ Graph (if (AND Lattice RootNodes)
                then (LAYOUTGRAPH Lattice RootNodes (SUBST 'LATTICE NC.*Graph*BrowserFormat
                                                           BrowserFormat)
                                  (fetch (SPECIALBROWSERSPECS Font) of SpecialBrowserSpecs)
                                  (fetch (SPECIALBROWSERSPECS MotherD) of SpecialBrowserSpecs)
                                  (fetch (SPECIALBROWSERSPECS PersonalD) of SpecialBrowserSpecs)
                                  (fetch (SPECIALBROWSERSPECS FamilyD) of SpecialBrowserSpecs))
              else (create GRAPH)))
(NC.SetBrowserLinksLegend Card (NC.MakeLinksLegend Graph Window DropVirtualNodesFlg))
(OR NoDisplayFlg (NC.PrintMsg InterestedWindow NIL "Done!"))
(NC.SetSubstance Card Graph)
(NC.SetBrowserLinkLabels Card (OR ListOfLinkLabels (LIST NC.SubBoxLinkLabel)))
(NC.SetBrowserRoots Card RootCards)
(NC.SetBrowserFormat Card BrowserFormat)
(NC.SetBrowserDepth Card Depth)
(NC.SetSpecialBrowserSpecs Card SpecialBrowserSpecs)
(NC.SetUserDataProp Card 'ReadOnly (NC.CardReadOnlyOpenP Card))
(COND
   (NoDisplayFlg (RETURN Card)))
(WINDOWPROP Window 'GRAPH Graph)
(NC.InstallTitleBarLeftMenu Window CardType)
(NC.InstallTitleBarMiddleMenu Window CardType)
(NC.RelayoutBrowserCard Window)
(WINDOWPROP Window 'RIGHTBUTTONFN (FUNCTION NC.BrowserRightButtonFn))
(WINDOWADDPROP Window 'SHRINKFN (FUNCTION NC.GraphCardShrinkFn))
```

;; SHOWGRAPH changes this REPAINTFN to REDISPLAYGRAPH, but since NC.RelayoutBrowserCard (now) only preserves
;; props that were already there, it has to be added here.

```
(WINDOWPROP Window 'REPAINTFN (FUNCTION REDISPLAYGRAPH))
(WINDOWADDPROP Window 'REPAINTFN (FUNCTION NC.BrowserRepaintFn)
      T)
(WINDOWPROP Window 'SCROLLFN (FUNCTION NC.BrowserScrollFn))
(WINDOWPROP Window 'RESHAPEFN (FUNCTION NC.BrowserReshapeFn))
(RETURN Window])
```

(NC.BringUpBrowserCard

```
[LAMBDA (Card Substance Region/Position)                    ; Edited 11-May-88 23:51 by Trigg
```

;; Given a browser Substance, open a browser window and set it up to be a NoteCard with ID.

;; rht 11/17/84: Now returns window.

;; rht 9/11/85: Now checks for changed link icon display global params.

;; rht 11/17/85: Now handles new card and Notefile objects.

;; rht 2/1/86: Now restores any saved UID user data info stashed on card's prop list.

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

;; rht 2/14/86: Now rebuilds browser hash array.

;; rht 2/28/86: Added WINDOWPROP for SCROLLFN and RESHAPEFN.

;; rht 3/2/86: Took out call to NC.FetchBrowserHashArray.

;; rht 4/5/86: Now only replaces graphnodes' TONODES' NODEID and DESTNODEID if they're non-nil.

;; rht 5/5/86: Took out call to NC.SetupTitleBarMenu.

```
    ;; rht&pmi 2/6/87: Moved call to NC.GraphLinkIconUpdateCheck in front of call to GraphCard's EditFn so as to remove 'double display' problem.

    ;; rht 1/16/88: Had to replace the ApplySuperTypeFn call with grungy apply* to prevent infinite recursive calls.

    ;; rht 5/11/88: Sped up things significantly by ripping out the conversion of old UIDs to new UID objects.  This enabled by changes to reading of
    ;; UIDs so that all versions of same UID read in will be eq.

    (DECLARE (GLOBALVARS NC.GlobalUIDHashArray))
    (LET ((GraphNodes (fetch (GRAPH GRAPHNODES) of Substance))
          Window)

        ;; Restore any saved UID user data info stashed on card UID's prop list.

        [for BrowserSavedLinkingInfoForNode in (NC.FetchBrowserSavedLinkingInfo Card)
           do (LET ((SourceUID (CAR BrowserSavedLinkingInfoForNode)))
                   (for SavedLinkingInfo on (CDR BrowserSavedLinkingInfoForNode) by (CDDR SavedLinkingInfo)
                      eachtime (BLOCK) do (NC.GraphNodeIDPutProp SourceUID (CAR SavedLinkingInfo)
                                                                 (CADR SavedLinkingInfo]
        (NC.SetBrowserSavedLinkingInfo Card NIL)

        ;; For each graph node corresponding to a notecard, hang the card object off the node id's prop list.

        [for GraphNode in GraphNodes bind LinkIcon DestCard eachtime (BLOCK)
           when (NC.LinkIconImageObjP (SETQ LinkIcon (fetch (GRAPHNODE NODELABEL) of GraphNode)))
           do (NC.GraphNodeIDPutProp (NC.CoerceToGraphNodeID GraphNode)
                     'CardObject
                     (SETQ DestCard (fetch (Link DestinationCard) of (NC.FetchLinkFromLinkIcon LinkIcon]

        ;; Make a new browser hash array with the new graph node UIDs.

        (NC.SetUserDataProp Card 'BrowserHashArray NIL)
        (NC.GetBrowserHashArray Card Substance)

        ;; For each graph node, fix the NODEID and DESTNODEID fields of each of its TONODES LinkParameters.  While we're at it, smash the
        ;; entry in the global uid hash array for each node's uid.

        [for GraphNode in GraphNodes eachtime (BLOCK)
           do (LET ((ThisNodeID (NC.CoerceToGraphNodeID GraphNode)))
                   (if (type? UID ThisNodeID)
                       then (PUTHASH ThisNodeID NIL NC.GlobalUIDHashArray))
                   (for ToNode in (fetch (GRAPHNODE TONODES) of GraphNode) eachtime (BLOCK)
                      when (EQ (CAR ToNode)
                               LINKPARAMS)
                      do (AND (LISTGET ToNode 'NODEID)
                              (LISTPUT ToNode 'NODEID ThisNodeID))
                         (AND (LISTGET ToNode 'DESTNODEID)
                              (LISTPUT ToNode 'DESTNODEID (NC.CoerceToGraphNodeID (CADR ToNode]

        ;; Bring up card and mess with its window.

        (NC.GraphLinkIconUpdateCheck Card NIL Substance T)
        (SETQ Window (APPLY* (NCP.CardTypeFn 'Graph 'EditFn)
                             Card Substance Region/Position))

        ;; I have to hang notecard's Card on window now in case REDISPLAYW runs and tries to get Card from window.

        (WINDOWPROP Window 'NoteCardObject Card)
        (NC.MakeLinksLegendMenu Window (NC.FetchBrowserLinksLegend Card))
                                                                    ; Disable the old-style right button grapher editor menu.
        (WINDOWPROP Window 'RIGHTBUTTONFN (FUNCTION NC.BrowserRightButtonFn))
        (WINDOWADDPROP Window 'SHRINKFN (FUNCTION NC.GraphCardShrinkFn))
        (WINDOWADDPROP Window 'REPAINTFN (FUNCTION NC.BrowserRepaintFn)
               T)
        (WINDOWPROP Window 'SCROLLFN (FUNCTION NC.BrowserScrollFn))
        (WINDOWPROP Window 'RESHAPEFN (FUNCTION NC.BrowserReshapeFn))

        ;; Check if link icon display global params have changed since last time card was up.  If so, fix graph nodes and redisplay.
                                                                    ; if (NC.GraphLinkIconUpdateCheck Card Window Substance T)
                                                                    ; then (REDISPLAYW Window)

        Window])

)

(DEFINEQ

(NC.GrowLinkLattice
  [LAMBDA (RootCardsList CurrentGraph ListOfLinkLabels GraphCard RemainingSearchDepth)
                                                                    (* pmi%: "30-Nov-87 12:48")

    (* Grow a lattice by following the links from RootID card among ListOfLinkLabels.
    Lattice will be fed to LAYOUTGRAPH, so for each note card encountered by following the links just fill in the ID, LABEL and
    daughter IDs)

    (* * rht 8/3/84%: Changed so as to also follow from links if they are present
    (prefixed by "_") on ListOfLinkLabels.)

    (* * rht 10/4/84%: Now stores the link label on the prop list of the NODEID of the graph under the property name of the
    destination ID. This is so that links can be drawn with dashing depending on the link's label.)

    (* * rht 3/8/85%: Added RemainingSearchDepth arg to limit the lattice growth to given depth.)

    (* * rht 8/9/85%: Changed so that backward links are no longer stored as a separate link type.
    Rather they're told apart from forward links by being stored on the destination node's prop list.)

    (* * rht 4/4/85%: Now first arg can be either a root Card or an existing graphnode.
    If the latter, then we're expanding an existing graph below that node.
```

If the former than we're starting a new lattice.)

(* * rht 10/17/85%: Changed from a recursive depth-first algorithm to a loop-driven breadth-first alg.)

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

(* * rht 5/26/87%: Now tries to follow cross-file links.)

(* * rht 10/26/87%: Now deactivates cards at the end that we had to NC.GetLinks for.)

(* * rht&pmi 11/30/87%: No longer allows following backlinks if they go to this browser.)

```
(LET
 (CardsAndDepthsQueue CardsNeedingDeactivation)              (* Make the queue contain pairs of root Card and depth
                                                             remaining to search.)
 (SETQ CardsAndDepthsQueue (for Card in RootCardsList collect (CONS Card RemainingSearchDepth)))
                                                             (* Make it a TCONC list for fast appending to the end.)
 (SETQ CardsAndDepthsQueue (CONS CardsAndDepthsQueue (LAST CardsAndDepthsQueue)))

      (* * Do breadth-first search using the queue IDsAndDepthsQueue.)

 [for bind CardAndDepth Card RemainingSearchDepth ToLinks FromLinks DestinationIDs GraphNodeID GraphNode
    eachtime (BLOCK)                                         (* Grab and take apart 1st pair on queue.)
           (SETQ CardAndDepth (CAAR CardsAndDepthsQueue))
           (SETQ Card (CAR CardAndDepth))
           (SETQ RemainingSearchDepth (CDR CardAndDepth))    (* Remove the front pair from the queue.)
           (RPLACA CardsAndDepthsQueue (CDAR CardsAndDepthsQueue))
                                                             (* If that was the last pair, then start queue over fresh.)
           (if (NULL (CAR CardsAndDepthsQueue))
               then (SETQ CardsAndDepthsQueue NIL))
    while Card unless (NC.SameCardP Card GraphCard)
    do
    (SETQ GraphNodeID (NC.GetBrowserNodeID GraphCard Card))  (* Go grab this ID's links.)
    (if (NC.ActiveCardP Card)
        then (SETQ ToLinks (NC.FetchToLinks Card))
             (SETQ FromLinks (NC.FetchFromLinks Card))
      else (NC.GetLinks Card)
           (SETQ ToLinks (NC.FetchToLinks Card))
           (SETQ FromLinks (NC.FetchFromLinks Card))
           (push CardsNeedingDeactivation Card))
    (if (IGREATERP RemainingSearchDepth 0)
        then                                                 (* Crush the ID's proplist.)
        (if (NOT (NC.GraphNodeIDGetProp GraphNodeID 'TouchedFlg))
            then (NC.SmashGraphNodeIDProps GraphNodeID)
                 (NC.GraphNodeIDPutProp GraphNodeID 'TouchedFlg T))
        (SETQ DestinationIDs
         (NCONC (for Link in ToLinks bind DestID DestVisitedFlg DestTouchedFlg ThisWayLinkFlg OtherWayLinkFlg
                   eachtime (BLOCK)
                          (if (SETQ ThisWayLinkFlg (NC.LinkLabelP Link ListOfLinkLabels))
                              then [SETQ DestID (NC.GetBrowserNodeID GraphCard
                                                   (LET ((DestCard (fetch (Link DestinationCard)
                                                                      of Link)))
                                                        (if (NC.CrossFileLinkCardP DestCard)
                                                            then (OR (NC.GetCrossFileLinkDestCard DestCard
                                                                     )
                                                                     DestCard)
                                                            else DestCard]
                                   (SETQ DestVisitedFlg (NC.GraphNodeIDGetProp DestID 'VisitedFlg))
                                   (SETQ DestTouchedFlg (NC.GraphNodeIDGetProp DestID 'TouchedFlg))
                                   (SETQ OtherWayLinkFlg (NC.ReverseLinkLabelP Link ListOfLinkLabels)))
                   when ThisWayLinkFlg unless (AND DestVisitedFlg OtherWayLinkFlg)
                   collect                                   (* Record presence of this link.)
                          (NC.UIDAddProp GraphNodeID DestID (fetch (Link Label) of Link)
                                 T)
                          DestID)
                 (for Link in FromLinks bind DestID DestTouchedFlg DestVisitedFlg ThisWayLinkFlg
                                             OtherWayLinkFlg LinkFromUsFlg SourceCard
                   eachtime (BLOCK)
                          (if [AND (SETQ ThisWayLinkFlg (NC.ReverseLinkLabelP Link ListOfLinkLabels))
                                   (NOT (SETQ LinkFromUsFlg (NC.SameCardP
                                                              GraphCard
                                                              (SETQ SourceCard
                                                               (LET ((SrcCard (fetch (Link SourceCard)
                                                                                 of Link)))
                                                                    (if (NC.CrossFileLinkCardP SrcCard)
                                                                        then (OR (NC.GetCrossFileLinkDestCard
                                                                                   SrcCard)
                                                                                 SrcCard)
                                                                        else SrcCard]
                              then (SETQ DestID (NC.GetBrowserNodeID GraphCard SourceCard))
                                   (SETQ DestVisitedFlg (NC.GraphNodeIDGetProp DestID 'VisitedFlg))
                                   (SETQ DestTouchedFlg (NC.GraphNodeIDGetProp DestID 'TouchedFlg))
                                   (SETQ OtherWayLinkFlg (NC.LinkLabelP Link ListOfLinkLabels)))
                   when ThisWayLinkFlg unless (OR LinkFromUsFlg (AND DestVisitedFlg OtherWayLinkFlg))
                   collect
```

(* Crush the dest node's prop list if it's never been touched. But if dest node is a fringe node for this search, don't have to

clear the whole proplist.)

```
                        (if (NOT DestTouchedFlg)
                            then (if (EQ 1 RemainingSearchDepth)
                                     then (NC.GraphNodeIDRemProp DestID GraphNodeID)
                                    else (NC.SmashGraphNodeIDProps DestID)
                                         (NC.GraphNodeIDPutProp DestID 'TouchedFlg T)))
                                                          (* Record presence of this link.)
                        (NC.UIDAddProp DestID GraphNodeID (fetch (Link Label) of Link)
                               T)
                        DestID)))
      (SETQ DestinationIDs (DREMOVE (NC.GetBrowserNodeID GraphCard GraphCard)
                                    (INTERSECTION DestinationIDs DestinationIDs)))
  else (SETQ DestinationIDs NIL))
(NC.GraphNodeIDPutProp GraphNodeID 'VisitedFlg T)
```

(* * Create new node and add to graph unless we're working on a node already in the graph.)

```
[if (SETQ GraphNode (FASSOC GraphNodeID CurrentGraph))
    then
```

(* If node is in graph, but we won't expand further, then leave it's destination IDs alone.)

```
       (AND (GREATERP RemainingSearchDepth 0)
            (replace (GRAPHNODE TONODES) of GraphNode with DestinationIDs))
  else (SETQ CurrentGraph (NCONC CurrentGraph
                                 (LIST (create GRAPHNODE
                                               NODEID _ GraphNodeID
                                               TONODES _ DestinationIDs
                                               NODELABEL _ Card]
```

(* * Attach new IDs to end of queue.)

```
(for DestinationID in DestinationIDs bind DestCard eachtime (BLOCK)
                                                    (SETQ DestCard (NC.CardFromBrowserNodeID
                                                                       DestinationID))
  unless [OR (NC.GraphNodeIDGetProp DestinationID 'VisitedFlg)
             (for CardAndDepth in (CAR CardsAndDepthsQueue) eachtime (BLOCK)
                thereis (NC.SameCardP DestCard (CAR CardAndDepth]
  do (SETQ CardsAndDepthsQueue (TCONC CardsAndDepthsQueue (CONS DestCard (SUB1 RemainingSearchDepth]
(for Card in CardsNeedingDeactivation do (NC.DeactivateCard Card))
CurrentGraph])
```

(NC.UpdateBrowserCard

```
[LAMBDA (Window)                                                    ; Edited 29-Jul-88 20:26 by Trigg
```

;; rht 10/14/84: Added call to DETACHALLWINDOWS to close any existing links legend window and prompt window. Also added call to ;; NC.MakeLinksLegend to make a new attached legend menu.

;; rht 1/15/85: Put hooks for AddNode, AddLink, etc. so editing graph edits underlying structure.

;; rht 2/14/85: Added ability to respecify roots and link labels before recomputing graph.

;; rht 3/8/85: Modified to use new browser props stored on card's proplist as of release 1.2.

;; rht 3/17/85: Now takes OnlyLayoutFlg argument. If set, then don't recompute lattice or ask about root nodes.

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

;; kirk 23Jan86 Changed to use NC.AskYesOrNo

;; rht 2/7/86: Now gets and sets browser format, etc. via fetch/set fns.

;; rht 3/7/86: Now only closes the Links legend menu attached window.

;; rht 6/10/86: Moved code to delete links legend menu and code to make new browser hash array to after questioning user about respecifying ;; roots.

;; rht 11/1/86: Added NC.ProtectedCardOperation wrapper and check for ops in progress.

;; pmi 12/5/86: Modified message to NC.SelectNoteCards to mention SHIFT-selection.

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

;; rht 12/16/86: Now checks that NC.MakeLink succeeded before creating a real link icon. If not, then make a standin for a cross file link icon.

;; rg 3/4/87 rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg

;; rg 3/18/87 added NC.CardSelectionOperation wrapper

;; rht 3/19/87: Fixed the part that calls NC.MakeLink so it really only rebuilds links if they've changed.

;; rg 4/1/87 changed CANCELLED to DON'T

;; rht 5/26/87: Now handles cross-file links properly, i.e. uses cross-file link standin in cases when GrowLinkLattice wasn't able to follow into the ;; remote notefile.

;; rht 7/29/88: Replaced code that lets user respecify roots by call to NC.RespecifyBrowserRoots. Also no longer calls NC.SetPropListDirtyFlg

```
(LET
 ((Card (NC.CoerceToCard Window)))
 (NC.ProtectedCardOperation
  Card "Recompute Browser Card" NIL
  (NCP.WithLockedCards
   (PROG (LinkLabels RootCards RootNodes Lattice LinkIcon Graph GraphNodes NodeLabel BrowserSpecs
              BrowserFormat DropVirtualNodesFlg Depth SpecialBrowserSpecs OldLabelNodes OldRootCards)
         (SETQ RootCards (NC.FetchBrowserRoots Card))
```

```
(NC.IfAllCardsFree
 (NC.LockListOfCards RootCards "Update Browser Card")
 (SETQ LinkLabels (NC.FetchBrowserLinkLabels Card))
 [SETQ BrowserFormat (OR (NC.FetchBrowserFormat Card)
                         '(LATTICE]                    ; If user wants *GRAPH* format, i.e. virtual nodes eliminated,
                                                       ; then set the flag
 (if (FMEMB NC.*Graph*BrowserFormat BrowserFormat)
     then (SETQ DropVirtualNodesFlg T))
 (SETQ Depth (OR (NC.FetchBrowserDepth Card)
                 999999))
 (SETQ SpecialBrowserSpecs (OR (NC.FetchSpecialBrowserSpecs Card)
                               (create SPECIALBROWSERSPECS)))
 [SETQ GraphNodes (fetch (GRAPH GRAPHNODES) of (SETQ Graph (WINDOWPROP Window 'GRAPH]
                                                       ; Get new roots.
 [if (OR (NULL RootCards)
         (NC.AskYesOrNo "Want to respecify roots? " "--" "No" T Window T NIL))
     then (SETQ RootCards (NC.RespecifyBrowserRoots Card RootCards GraphNodes Window))
          (COND
             ((EQ RootCards 'DON'T)
              (RETURN]                                 ; Get rid of the links legend menu attached window.
 (for Win in (ATTACHEDWINDOWS Window) when (WINDOWPROP Win 'LINKSLEGENDWINP)
    do (DETACHWINDOW Win)
       (CLOSEW Win))                                   ; Smash the current hash array, putting a fresh one in its place.
 (NC.GetBrowserHashArray Card)
 (NC.PrintMsg Window T (CHARACTER 13)
        "Computing browser graph. Please wait. ...")
                                                       ; Compute lattice breadth-first from the roots.
 (SETQ Lattice (NC.GrowLinkLattice RootCards NIL LinkLabels Card Depth))
 (SETQ RootNodes (for RootCard in RootCards collect (NC.GetBrowserNodeID Card RootCard)))
                                                       ; Remove all links that are in the old browser graph but not in the
                                                       ; new one
 [for Node in GraphNodes eachtime (BLOCK) unless [for LatticeNode in Lattice
                                                    bind (CardForNode _ (NC.CardFromBrowserNodeID
                                                                          (NC.CoerceToGraphNodeID Node)
                                                                          ))
                                                    thereis (NC.SameCardP CardForNode
                                                                  (NC.CardFromBrowserNodeID
                                                                   (NC.CoerceToGraphNodeID LatticeNode
                                                                          ]
    do (LET ((NodeLabel (fetch (GRAPHNODE NODELABEL) of Node)))
            (COND
               ((NC.LinkIconImageObjP NodeLabel)
                (NC.DeleteLink (NC.FetchLinkFromLinkIcon NodeLabel)
                       T T))
               ((STRINGP NodeLabel)                    ; Collect the label nodes from the old browser.
                (SETQ OldLabelNodes (CONS Node OldLabelNodes]
                                                       ; Create Links for all nodes in the new browser graph but not in
                                                       ; the old one.
 [for Node in Lattice eachtime (BLOCK) bind (CrossFileLinkModePropList _ (LIST (fetch (Card NoteFile)
                                                                                    of Card)
                                                                              NIL))
    do (LET [(NodeID (fetch (GRAPHNODE NODEID) of Node))
             (OldNode (for GraphNode in GraphNodes bind (CardForNode _ (NC.CardFromBrowserNodeID
                                                                         (NC.CoerceToGraphNodeID Node)
                                                                         ))
                               when (NC.SameCardP CardForNode (NC.CardFromBrowserNodeID (
                                                                   NC.CoerceToGraphNodeID
                                                                         GraphNode)))
                               do (RETURN GraphNode]
            [if OldNode
                then (replace (GRAPHNODE NODELABEL) of Node with (fetch (GRAPHNODE NODELABEL)
                                                                        of OldNode))
              else (replace (GRAPHNODE NODELABEL) of Node
                      with (LET (NewLink)
                                (if [AND (NOT (NC.CrossFileLinkCardP (fetch (GRAPHNODE NODELABEL)
                                                                            of Node)))
                                         (SETQ NewLink (NC.MakeLink Window
                                                                    NC.BrowserContentsLinkLabel
                                                                    (fetch (GRAPHNODE NODELABEL)
                                                                       of Node)
                                                                    Card NIL NIL NIL NIL NIL
                                                                    (NC.ComputeCrossFileLinkMode
                                                                     (fetch (GRAPHNODE NODELABEL)
                                                                        of Node)
                                                                     CrossFileLinkModePropList Window]
                                    then (NC.MakeLinkIcon NewLink)
                                  else (NC.MakeCrossFileLinkIconStandIn (fetch (GRAPHNODE NODELABEL)
                                                                               of Node]
                                                       ; Untouch each graph node so that next Recompute will put fresh
                                                       ; values on proplist.
            (NC.GraphNodeIDRemProp NodeID 'TouchedFlg)
            (NC.GraphNodeIDRemProp NodeID 'VisitedFlg]
                                                       ; Throw in the label nodes from the old browser.
 (SETQ Lattice (NCONC Lattice OldLabelNodes))
 ;; For each old label node, take away nonexistent fromnodes and save the label nodes that no longer have any from nodes.
 (for OldLabelNode in OldLabelNodes eachtime (BLOCK)
```

```
        do (replace (GRAPHNODE FROMNODES) of OldLabelNode
              with (for FromNodeID in (fetch (GRAPHNODE FROMNODES) of OldLabelNode) bind FromNode
                      eachtime (BLOCK) when (SETQ FromNode (FASSOC FromNodeID Lattice))
                      collect                                   ; If the From node isn't a label node, then add to its Tonode list.
                             [if (NC.LinkIconImageObjP (fetch (GRAPHNODE NODELABEL) of FromNode))
                                 then (replace (GRAPHNODE TONODES) of FromNode
                                         with (CONS (fetch (GRAPHNODE NODEID) of OldLabelNode)
                                                    (fetch (GRAPHNODE TONODES) of FromNode]
                             FromNodeID))                       ; For the old label node's ToNodes, just need to remove any for
                                                                ; ToNodes that no longer exist.
           (replace (GRAPHNODE TONODES) of OldLabelNode
              with (for ToNodeID in (fetch (GRAPHNODE TONODES) of OldLabelNode) bind ToNode
                      eachtime (BLOCK) when (SETQ ToNode (FASSOC ToNodeID Lattice))
                      collect                                   ; If the To node isn't a label node, then add to its FromNode list.
                             [if (NC.LinkIconImageObjP (fetch (GRAPHNODE NODELABEL) of ToNode))
                                 then (replace (GRAPHNODE FROMNODES) of ToNode
                                         with (CONS (fetch (GRAPHNODE NODEID) of OldLabelNode)
                                                    (fetch (GRAPHNODE FROMNODES) of ToNode]
                             ToNodeID)))

     ;; Layout graph, including as roots any non-virtual nodes with no from nodes to avoid disconnected graphs.

     (SETQ Graph (if (AND Lattice RootNodes)
                     then (LAYOUTGRAPH Lattice (for Node in Lattice bind NodeID
                                                  eachtime (BLOCK)
                                                         (SETQ NodeID (OR (NC.CoerceToGraphNodeID Node)
                                                                          (fetch (GRAPHNODE NODEID)
                                                                             of Node)))
                                                  when (OR (FMEMB NodeID RootNodes)
                                                           (NULL (fetch (GRAPHNODE FROMNODES)
                                                                    of Node)))
                                                  collect NodeID)
                                               (SUBST 'LATTICE NC.*Graph*BrowserFormat BrowserFormat)
                                               (fetch (SPECIALBROWSERSPECS Font) of SpecialBrowserSpecs)
                                               (fetch (SPECIALBROWSERSPECS MotherD) of SpecialBrowserSpecs)
                                               (fetch (SPECIALBROWSERSPECS PersonalD) of SpecialBrowserSpecs)
                                               (fetch (SPECIALBROWSERSPECS FamilyD) of SpecialBrowserSpecs))
                   else (create GRAPH)))                        ; Build links legend and fix up TONODES in the graph.
     (NC.SetBrowserLinksLegend Card (NC.MakeLinksLegend Graph Window DropVirtualNodesFlg))
     (NC.SetBrowserRoots Card RootCards)
     (NC.SetBrowserDepth Card Depth)
     (WINDOWPROP Window 'GRAPH Graph)
     (NC.RelayoutBrowserCard Window])
```

(NC.RelayoutBrowserCard

```
  [LAMBDA (Window)                                              ; Edited 28-Sep-88 18:45 by jrc
```

;;; Called from the middle button of a browser or structeditbrowser card. This lays out and displays the browser, but does not recompute the nodes.

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

;; rht 2/7/86: Now gets browser format, etc. via fetch/set fns.

;; rht 2/28/86: Added WINDOWPROP for SCROLLFN and RESHAPEFN.

;; rht 5/8/86: Added calls to rig title bar properly.

;; fgh 6/30/86 Added NC.GRAPHERCOPYBUTTONEVENTFN to SHOWGRAPH call

;; rht 11/1/86: Added NC.ProtectedCardOperation wrapper and check for ops in progress.

;; rg 3/4/87 rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg

;; rht 7/29/88: No longer calls NC.SetPropListDirtyFlg

;; jrc 28-sep-88: Saves various WINDOWPROPs that SHOWGRAPH seems to smash. The original code made assumptions about what the value
;; of each of the WINDOWPROPs was.

```
(LET ((Card (NC.CoerceToCard Window)))
     (NC.ProtectedCardOperation
      Card "Relayout Browser Card" NIL
      (PROG (RootCards RootNodeIDs OldToNodePairs Graph GraphNodes BrowserFormat DropVirtualNodesFlg
             SpecialBrowserSpecs OldRepaintFn OldRightButtonFn OldScrollFn OldReshapeFn)
            (NC.PrintMsg Window T "Laying out graph ...")
            (SETQ RootCards (NC.FetchBrowserRoots Card))
            [SETQ BrowserFormat (OR (NC.FetchBrowserFormat Card)
                                    'LATTICE]                   ; If user wants *GRAPH* format, i.e. virtual nodes eliminated,
                                                                ; then set the flag
            (if (FMEMB NC.*Graph*BrowserFormat BrowserFormat)
                then (SETQ DropVirtualNodesFlg T))
            (SETQ SpecialBrowserSpecs (OR (CAR (NC.FetchSpecialBrowserSpecs Card))
                                          (create SPECIALBROWSERSPECS)))
            [SETQ GraphNodes (fetch (GRAPH GRAPHNODES) of (SETQ Graph (WINDOWPROP Window 'GRAPH]
                                                                ; Create hash array if haven't already.
            (NC.GetBrowserHashArray Card Graph)                 ; check graph node size against image box size.
            (NC.GraphLinkIconUpdateCheck Card Window Graph NIL)
```

;; Save the TONODES values of the nodes so can replace later after LAYOUTGRAPH call. At the same time, throw away all the link
;; params info in TONODES field.

```
            [SETQ OldToNodePairs (for Node in GraphNodes bind ToNodes eachtime (BLOCK)
                                    collect (PROG1 [CONS (fetch (GRAPHNODE NODEID) of Node)
                                                         (APPEND (SETQ ToNodes (fetch (GRAPHNODE TONODES)
```

```
                                                                    of Node]
                                                          (replace (GRAPHNODE TONODES) of Node
                                                             with (for ToNode in ToNodes
                                                                      collect (if (EQ (CAR ToNode)
                                                                                      LINKPARAMS)
                                                                                  then (CADR ToNode)
                                                                                else ToNode)))]
                    (SETQ RootNodeIDs (for RootCard in RootCards collect (NC.GetBrowserNodeID Card RootCard)))
```

;; Layout graph, including as roots any non-virtual nodes with no from nodes to avoid disconnected graphs.

```
                    (SETQ Graph (if GraphNodes
                                    then (LAYOUTGRAPH GraphNodes
                                                      (for Node in GraphNodes bind NodeID eachtime (BLOCK)
                                                                                                (SETQ NodeID
                                                                                                 (fetch (GRAPHNODE NODEID)
                                                                                                    of Node))
                                                         when (OR (AND (NULL (fetch (GRAPHNODE FROMNODES) of Node))
                                                                       (NOT (LISTP NodeID)))
                                                                  (FMEMB NodeID RootNodeIDs))
                                                         collect NodeID)
                                                      (SUBST 'LATTICE NC.*Graph*BrowserFormat BrowserFormat)
                                                      (fetch (SPECIALBROWSERSPECS Font) of SpecialBrowserSpecs)
                                                      (fetch (SPECIALBROWSERSPECS MotherD) of SpecialBrowserSpecs)
                                                      (fetch (SPECIALBROWSERSPECS PersonalD) of SpecialBrowserSpecs)
                                                      (fetch (SPECIALBROWSERSPECS FamilyD) of SpecialBrowserSpecs))
                                  else (create GRAPH)))
```

;; Replace the TONODES fields of the Graph nodes by their pre-LAYOUTGRAPH values. Also throw away any nodes that didn't appear
;; in the old graph.

```
                    (if Graph
                        then (replace (GRAPH GRAPHNODES) of Graph
                                  with (for Node in (fetch (GRAPH GRAPHNODES) of Graph) bind AssocPair eachtime (BLOCK)
                                          when (SETQ AssocPair (FASSOC (fetch (GRAPHNODE NODEID) of Node)
                                                                       OldToNodePairs))
                                          collect (replace (GRAPHNODE TONODES) of Node with (CDR AssocPair))
                                                  (if DropVirtualNodesFlg
                                                      then (replace (GRAPHNODE NODEBORDER) of Node with NIL))
                                                                       ; Throw away LINKPARAMS junk from the FromNodes that
                                                                       ; LAYOUTGRAPH stuck in.
                                                  (replace (GRAPHNODE FROMNODES) of Node
                                                     with (for FromNode in (fetch (GRAPHNODE FROMNODES) of Node)
                                                             eachtime (BLOCK) collect (if (EQ (CAR FromNode)
                                                                                              LINKPARAMS)
                                                                                          then (CADR FromNode)
                                                                                        else FromNode)))
                                                 Node)))
```

;;; have to preserve windowprops since SHOWGRAPH messes with them

```
                    (SETQ OldRepaintFn (WINDOWPROP Window 'REPAINTFN))
                    (SETQ OldRightButtonFn (WINDOWPROP Window 'RIGHTBUTTONFN))
                    (SETQ OldScrollFn (WINDOWPROP Window 'SCROLLFN))
                    (SETQ OldReshapeFn (WINDOWPROP Window 'RESHAPEFN))
                    (SHOWGRAPH Graph Window (FUNCTION NC.GraphCardLeftButtonFn)
                           (FUNCTION NC.GraphCardMiddleButtonFn)
                           NIL T (FUNCTION NC.GRAPHERCOPYBUTTONEVENTFN))
```

;;; Have to reset windowprops since SHOWGRAPH messes with them.

```
                                                                    ; Disable the old-style right button grapher editor menu.
                    (WINDOWPROP Window 'RIGHTBUTTONFN OldRightButtonFn)
                    (WINDOWPROP Window 'REPAINTFN OldRepaintFn)
                    (WINDOWPROP Window 'SCROLLFN OldScrollFn)
                    (WINDOWPROP Window 'RESHAPEFN OldReshapeFn)
                    (NC.SetSubstance Card (WINDOWPROP Window 'GRAPH))
                    (NC.MarkCardDirty Card)
                    (NC.InstallTitleBarButtonEventFn Window (FUNCTION NC.TitleBarButtonEventFn))
                    (NC.InstallCopyButtonEventFn Window)
                    (NC.ClearMsg Window T])
```

(NC.LayoutNewBrowserNodes

```
  [LAMBDA (RootNode NewNodes BrowserFormat SpecialBrowserSpecs)  ; Edited 27-Jul-90 09:20 by tafel
```

(* * Hold onto old location of RootNode. Then layout the subgraph having root RootNode and lattice NewNodes.
Finally, translate the locations of NewNodes using old loc of RootNode.)

(* * rht 8/21/86%: LAYOUTGRAPH destroys FROMNODES of root node.
Now we save and restore these after LAYOUTGRAPH is called.)

(* * rht 11/5/86%: Now replaces ToNodes of each node with a subset consisting only of nodes in NewNodes so
LAYOUTGRAPH won't break. Afterwards, puts the original list back.)

```
    (DECLARE (GLOBALVARS NC.*Graph*BrowserFormat))
    (LET ((OldRootNodePos (fetch (GRAPHNODE NODEPOSITION) of RootNode))
          (Lattice (CONS RootNode NewNodes))
          (FromNodes (fetch (GRAPHNODE FROMNODES) of RootNode))
          NewRootNodePos)                                       (* Stash old ToNodes and replace with intersection of original
                                                                ToNodes and NewNodes.)
```

```
        [for Node in Lattice do (LET ((NodeID (NC.CoerceToGraphNodeID Node))
                                      (ToNodes (fetch (GRAPHNODE TONODES) of Node)))
                                     (NC.GraphNodeIDPutProp NodeID 'OldToNodes ToNodes)
                                     (replace (GRAPHNODE TONODES) of Node with (for Node in ToNodes
                                                                                  when (FASSOC Node Lattice)
                                                                                  collect Node]
        (LAYOUTGRAPH Lattice (LIST (fetch (GRAPHNODE NODEID) of RootNode))
               (SUBST 'LATTICE NC.*Graph*BrowserFormat BrowserFormat)
               (fetch (SPECIALBROWSERSPECS Font) of SpecialBrowserSpecs)
               (fetch (SPECIALBROWSERSPECS MotherD) of SpecialBrowserSpecs)
               (fetch (SPECIALBROWSERSPECS PersonalD) of SpecialBrowserSpecs)
               (fetch (SPECIALBROWSERSPECS FamilyD) of SpecialBrowserSpecs))
                                                                  (* Restore FROMNODES of RootNode.)
        (replace (GRAPHNODE FROMNODES) of RootNode with FromNodes)
        (SETQ NewRootNodePos (fetch (GRAPHNODE NODEPOSITION) of RootNode))
        (for Node in Lattice bind (XOffset _ (DIFFERENCE (fetch (POSITION XCOORD) of OldRootNodePos)
                                                         (fetch (POSITION XCOORD) of NewRootNodePos)))
                                  (YOffset _ (DIFFERENCE (fetch (POSITION YCOORD) of OldRootNodePos)
                                                         (fetch (POSITION YCOORD) of NewRootNodePos)))
          do                                                      (* Recover old list of ToNodes.)
             (replace (GRAPHNODE TONODES) of Node with (NC.GraphNodeIDGetProp (NC.CoerceToGraphNodeID Node)
                                                                              'OldToNodes))
             (LET ((NodePos (fetch (GRAPHNODE NODEPOSITION) of Node)))
                  (replace (GRAPHNODE NODEPOSITION) of Node with (create POSITION
                                                                         XCOORD _ (PLUS XOffset
                                                                                        (fetch (POSITION XCOORD)
                                                                                           of NodePos))
                                                                         YCOORD _ (PLUS YOffset
                                                                                        (fetch (POSITION YCOORD)
                                                                                           of NodePos])
```

(NC.ConnectNodesInBrowser

```
  [LAMBDA (Window)                                                (* rht%: "29-May-87 00:13")
```

(* * Draw any links, from the current link set, between any pairs of nodes currently being shown in the browser.)

(* * rht 2/7/86%: Now gets and sets browser format, etc. via fetch/set fns.
Also fixed one last old call to GETPROPLIST on a NodeID.)

(* * rht 3/2/86%: Added WINDOWPROP for SCROLLFN and RESHAPEFN.)

(* * fgh |5/21/86| Updated reinstallation of title bar menus after SHOWGRAPH to use new title bar menu mechanism.)

(* * rht 6/10/86%: Now calls NC.ShowBrowserGraph.)

(* * rht 11/1/86%: Added NC.ProtectedCardOperation wrapper and check for ops in progress.)

(* * rg |3/4/87| rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg)

(* * rht 5/28/87%: Modified to handle cross-file links.)

```
    (LET ((Card (NC.CoerceToCard Window)))
         (NC.ProtectedCardOperation
          Card "Reconnect Browser Card" NIL
          (PROG (LinkLabels Graph GraphNodes BrowserFormat DropVirtualNodesFlg NodeIDs)
                (SETQ LinkLabels (NC.FetchBrowserLinkLabels Card))
                (SETQ BrowserFormat (NC.FetchBrowserFormat Card))  (* If user wants *GRAPH* format, i.e.
                                                                   virtual nodes eliminated, then set the flag)
                (if (FMEMB NC.*Graph*BrowserFormat BrowserFormat)
                    then (SETQ DropVirtualNodesFlg T))
                [SETQ GraphNodes (fetch (GRAPH GRAPHNODES) of (SETQ Graph (WINDOWPROP Window 'GRAPH]
                                                                   (* Create hash array if haven't already.)
                (NC.GetBrowserHashArray Card Graph)                (* check graph node size against image box size.)
                (NC.GraphLinkIconUpdateCheck Card Window Graph NIL)
                                                                   (* These are the workhorse loops that rebuild the TONODES of
           each nonvirtual node.)                                  (* First smash all the nodeID's proplists and accumulate
                                                                   nodeIDs.)
                (SETQ NodeIDs (for Node in GraphNodes bind NodeID eachtime (BLOCK)
                                 when (NC.LinkIconImageObjP (fetch (GRAPHNODE NODELABEL) of Node))
                                 collect (NC.SmashGraphNodeIDProps (SETQ NodeID (NC.CoerceToGraphNodeID Node)))
                                       NodeID))                    (* Throw away duplicates obtained from virtual nodes.)
                (SETQ NodeIDs (INTERSECTION NodeIDs NodeIDs))
```

(* Next accumulate all linktypes on the from node's proplist using the To node's graphnodeID as prop name.
We do the analogous thing for backward links, but notice that we ignore backward linktypes that also appear in the list in their forward version.)

```
                (for NodeID in NodeIDs bind RealCard
                   do (for Link in (NC.RetrieveToLinks (SETQ RealCard (NC.CardFromBrowserNodeID NodeID)))
                         bind DestNodeID eachtime (BLOCK) when (NC.LinkLabelP Link LinkLabels)
                         when [LET ((DestCard (fetch (Link DestinationCard) of Link)))
                                   (SETQ DestNodeID (NC.GetBrowserNodeID Card
                                                                         (if (NC.CrossFileLinkCardP DestCard)
                                                                             then (OR (NC.GetCrossFileLinkDestCard DestCard
                                                                                                                   Window)
                                                                                      DestCard)
```

```
                                                    else DestCard)))
                                    (SETQ DestNodeID (for ID in NodeIDs when (NC.SameUIDP DestNodeID ID)
                                                        do (RETURN ID]
                                       do (NC.UIDAddProp NodeID DestNodeID (fetch (Link Label) of Link)
                                                 T))
                                    (for Link in (NC.RetrieveFromLinks RealCard) bind SourceNodeID eachtime (BLOCK)
                                       when (AND (NC.ReverseLinkLabelP Link LinkLabels)
                                                 (NOT (NC.LinkLabelP Link LinkLabels)))
                                       when [LET ((SourceCard (fetch (Link SourceCard) of Link)))
                                                 (SETQ SourceNodeID (NC.GetBrowserNodeID Card
                                                                         (if (NC.CrossFileLinkCardP SourceCard)
                                                                             then (OR (NC.GetCrossFileLinkDestCard
                                                                                          SourceCard Window)
                                                                                      SourceCard)
                                                                           else SourceCard)))
                                                 (SETQ SourceNodeID (for ID in NodeIDs when (NC.SameUIDP SourceNodeID ID)
                                                                       do (RETURN ID]
                                       do (NC.UIDAddProp SourceNodeID NodeID (fetch (Link Label) of Link)
                                                 T)))
                                    [for Node in GraphNodes bind NodeID OldToNodeIDs eachtime (BLOCK)
                                       unless (LISTP (SETQ NodeID (fetch (GRAPHNODE NODEID) of Node)))
                                       when (NC.LinkIconImageObjP (fetch (GRAPHNODE NODELABEL) of Node))
                                       do                                          (* Accumulate the old NodeIDs, possibly virtual, from the
                                                                                   TONODES list.)
                                          (SETQ OldToNodeIDs (for ToNode in (fetch (GRAPHNODE TONODES) of Node)
                                                                collect (if (EQ (CAR ToNode)
                                                                                LINKPARAMS)
                                                                            then (CADR ToNode)
                                                                          else ToNode)))

                                    (* The trick here is to use a virtual node for this ToNode if one was used before, otherwise just the ToNodeID.
                                    Also throw in the label nodes that were in the TONODES list before.)

                                          (replace (GRAPHNODE TONODES) of Node
                                             with (NCONC (for ToNodeID on (NC.ComputeBrowserSavedLinkingInfoForNode NodeID)
                                                            by (CDDR ToNodeID) eachtime (BLOCK)
                                                            collect (OR (for OldToNodeID in OldToNodeIDs
                                                                           thereis (AND (LISTP OldToNodeID)
                                                                                        (EQ (CAR ToNodeID)
                                                                                            (CAR OldToNodeID))
                                                                                        OldToNodeID))
                                                                        (CAR ToNodeID)))
                                                         (for ToNodeID in OldToNodeIDs eachtime (BLOCK)
                                                            unless (NC.SameCardP Card (NC.CardFromBrowserNodeID (
                                                                                                                NC.CoerceToGraphNodeID
                                                                                                                 ToNodeID)))
                                                            collect ToNodeID]
                                          (NC.RebuildFromNodesInGraph GraphNodes)
                                          (NC.SetBrowserLinksLegend Card (NC.MakeLinksLegend Graph Window DropVirtualNodesFlg))
                                                                                   (* Display the graph.)
                                          (NC.ShowBrowserGraph Graph Window)
                                          (NC.SetSubstance Card (WINDOWPROP Window 'GRAPH))
                                          (NC.MarkCardDirty Card)
                                          (NC.ClearMsg Window T])
```

(NC.UnconnectNodesInBrowser

```
  [LAMBDA (Window)                                                 (* Randy.Gobbel " 4-Mar-87 13:51")

          (* * Remove all the links in the browser.)

          (* * rht 11/17/85%: Now handles new card and notefile objects.)

          (* * rht 2/7/86%: Now gets and sets browser format, etc. via fetch/set fns.)

          (* * fgh |5/21/86| Updated reinstallation of title bar menus after SHOWGRAPH to use new title bar menu mechanism.)

          (* * rht 6/10/86%: Now calls NC.ShowBrowserGraph.)

          (* * rht 11/1/86%: Added NC.ProtectedCardOperation wrapper and check for ops in progress.)

          (* * rg |3/4/87| rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg)

    (LET ((Card (NC.CoerceToCard Window)))
         (NC.ProtectedCardOperation Card "Unconnect Browser Card" NIL
                (PROG (Graph GraphNodes BrowserFormat DropVirtualNodesFlg)
                      (SETQ BrowserFormat (NC.FetchBrowserFormat Card))
                                                                   (* If user wants *GRAPH* format, i.e.
                                                                   virtual nodes eliminated, then set the flag)
                      (if (FMEMB NC.*Graph*BrowserFormat BrowserFormat)
                          then (SETQ DropVirtualNodesFlg T))
                      [SETQ GraphNodes (fetch (GRAPH GRAPHNODES) of (SETQ Graph (WINDOWPROP Window 'GRAPH]
                                                                   (* smash all the nodeID's proplists and TONODES fields.)
                      (for Node in GraphNodes bind NodeID unless [PROGN (BLOCK)
                                                                        (LISTP (SETQ NodeID (fetch (GRAPHNODE NODEID
                                                                                                                  )
                                                                                                   of Node]
```

```
                    do (NC.SmashGraphNodeIDProps NodeID)
                       (replace (GRAPHNODE TONODES) of Node with NIL)
                       (replace (GRAPHNODE FROMNODES) of Node with NIL))
                 (NC.MakeLinksLegend Graph Window DropVirtualNodesFlg)
                                                          (* Display the graph.)
                 (NC.ShowBrowserGraph Graph Window)
                 (NC.SetSubstance Card (WINDOWPROP Window 'GRAPH))
                 (NC.MarkCardDirty Card)
                 (NC.SetBrowserLinksLegend Card NIL)
                 (NC.ClearMsg Window T])
```

(NC.ExpandBrowserNode

```
  [LAMBDA (Window)                                        ; Edited 29-Jul-88 20:25 by Trigg
```

;;; Ask user to choose a node in the browser and recompute the part of the lattice under that node to the given depth. And relayout the graph. The code ;;; is just a modification of the NC.UpdateBrowserCard code.

;; rht 2/7/86: Now gets and sets browser format, etc. via fetch/set fns.

;; rht 6/10/86: No longer does relayout after expand. Uses NC.LayoutNewBrowserNodes to compute proper locations of new nodes. Also calls ;; NC.ShowBrowserGraph.

;; rht 11/1/86: Added NC.ProtectedCardOperation wrapper and check for ops in progress.

;; rg 3/4/87 rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg

;; rht 5/26/87: Now handles cross-file links properly, i.e. uses cross-file link standin in cases when GrowLinkLattice wasn't able to follow into the ;; remote notefile.

;; rht 7/29/88: No longer calls NC.SetPropListDirtyFlg

```
(LET
 ((Card (NC.CoerceToCard Window)))
 (NC.ProtectedCardOperation
  Card "Expand Node of Browser Card" NIL
  (PROG (NodeToExpand LinkLabels RootCards RootNodes Lattice LinkIcon OldToNodePairs Graph GraphNodes
               NodeLabel OldNode Link BrowserSpecs BrowserFormat DropVirtualNodesFlg Depth
               SpecialBrowserSpecs SavedLabelNodes NewNodes)
        (SETQ RootCards (NC.FetchBrowserRoots Card))
        (SETQ LinkLabels (NC.FetchBrowserLinkLabels Card))
        (SETQ BrowserFormat (NC.FetchBrowserFormat Card))      ; If user wants *GRAPH* format, i.e. virtual nodes eliminated,
                                                               ; then set the flag
        (if (FMEMB NC.*Graph*BrowserFormat BrowserFormat)
            then (SETQ DropVirtualNodesFlg T))
        (SETQ SpecialBrowserSpecs (OR (NC.FetchSpecialBrowserSpecs Card)
                                      (create SPECIALBROWSERSPECS)))
        [SETQ GraphNodes (fetch (GRAPH GRAPHNODES) of (SETQ Graph (WINDOWPROP Window 'GRAPH]
                                                               ; If there aren't any nodes in graph, then get out pronto.
        (if (NULL GraphNodes)
            then (NC.PrintMsg Window T "No nodes to expand.")
                 (DISMISS 1000)
                 (NC.ClearMsg Window T)
                 (RETURN NIL))                                 ; Create hash array if haven't already.
        (NC.GetBrowserHashArray Card Graph)
        (NC.PrintMsg Window T "Pick node to expand." (CHARACTER 13))
                                                               ; Note call to the grapher function READ/NODE to select a graph
                                                               ; node.
        (SETQ NodeToExpand (READ/NODE GraphNodes Window))      ; Can't expand a label node.
        (if (NOT (NC.LinkIconImageObjP (fetch (GRAPHNODE NODELABEL) of NodeToExpand)))
            then (NC.PrintMsg NIL T "Sorry, can't expand a label node.")
                 (FLASHW PROMPTWINDOW)
                 (NC.ClearMsg Window T)
                 (RETURN))
        (SETQ Depth (MKATOM (NC.AskUser "Depth to expand (type integer or INF): " "--" 1 T Window NIL NIL T)
                            ))
        (COND
           ((EQ Depth 'INF)
            (SETQ Depth MAX.FIXP))
           ((NOT (AND (FIXP Depth)
                      (GREATERP Depth 0)))
            (NC.PrintMsg Window T "Depth must be an integer greater than 0 or INF.")
            (RETURN)))
        (NC.PrintMsg Window T (CHARACTER 13)
               "Augmenting browser graph. Please wait. ...")
```

;; Save the nodes pointed to by the chosen node that are label nodes. GrowLinkLattice will trash those, so we restore afterwards.

```
        (SETQ SavedLabelNodes (for ToNode in (fetch (GRAPHNODE TONODES) of NodeToExpand) eachtime (BLOCK)
                                 when (AND (NOT (EQ (CAR ToNode)
                                                    LINKPARAMS))
                                           (NOT (NC.LinkIconImageObjP ToNode)))
                                 collect ToNode))              ; Increase link lattice from chosen node to given depth.
        (SETQ Lattice (NC.GrowLinkLattice (LIST (NC.CardFromBrowserNodeID (fetch (GRAPHNODE NODEID)
                                                                                 of NodeToExpand)))
                                (APPEND GraphNodes)
                                LinkLabels Card Depth))
        [AND SavedLabelNodes (replace (GRAPHNODE TONODES) of NodeToExpand with (APPEND SavedLabelNodes
                                                                                       (fetch (GRAPHNODE TONODES
                                                                                                         )
                                                                                              of NodeToExpand]
        (SETQ RootNodes (for RootCard in RootCards collect (NC.GetBrowserNodeID Card RootCard)))
```

```
                                                                ; Create Links for all nodes in the new browser graph but not in
                                                                ; the old one.
          [for Node in Lattice bind NodeID (CrossFileLinkModePropList _ (LIST (fetch (Card NoteFile) of Card)
                                                                                    NIL))
             do (COND
                   ((SETQ OldNode (FASSOC (SETQ NodeID (OR (NC.CoerceToGraphNodeID Node)
                                                           (fetch (GRAPHNODE NODEID) of Node)))
                                          GraphNodes))
                    (replace (GRAPHNODE NODELABEL) of Node with (fetch (GRAPHNODE NODELABEL) of OldNode)))
                   (T [replace (GRAPHNODE NODELABEL) of Node
                         with (LET (NewLink)
                                   (if [AND (NOT (NC.CrossFileLinkCardP (fetch (GRAPHNODE NODELABEL) of Node)))
                                            (SETQ NewLink (NC.MakeLink Window NC.BrowserContentsLinkLabel
                                                                       (fetch (GRAPHNODE NODELABEL) of Node)
                                                                       Card NIL NIL NIL NIL NIL
                                                                       (NC.ComputeCrossFileLinkMode
                                                                        (fetch (GRAPHNODE NODELABEL) of Node)
                                                                        CrossFileLinkModePropList Window]
                                       then (NC.MakeLinkIcon NewLink)
                                     else (NC.MakeCrossFileLinkIconStandIn (fetch (GRAPHNODE NODELABEL)
                                                                                  of Node]
                                                                ; Make a list of all new nodes.
                      (push NewNodes Node)))                    ; Throw away virtual node info.
                (AND NodeID (replace (GRAPHNODE NODEID) of Node with NodeID))
                                                                ; Untouch each graph node so that next Recompute will put fresh
                                                                ; values on proplist.
                (NC.GraphNodeIDRemProp NodeID 'TouchedFlg)
                (NC.GraphNodeIDRemProp NodeID 'VisitedFlg)

                ;; Smash all the unnecessary junk off existing nodes, letting LAYOUTGRAPH and NC.MakeLinksLegend recompute.

                (replace (GRAPHNODE TONODES) of Node
                   with (for ToNode in (fetch (GRAPHNODE TONODES) of Node) bind ToNodeID eachtime (BLOCK)
                           collect (if (SETQ ToNodeID (NC.CoerceToGraphNodeID ToNode))
                                       then                     ; Throw away link parameterlist info.
                                                                ; Throw away link dashing info.
                                            (NC.GraphNodeIDPutProp NodeID ToNodeID
                                                   (for LabelPair in (NC.GraphNodeIDGetProp NodeID ToNodeID)
                                                      collect (OR (CAR LabelPair)
                                                                  LabelPair)))
                                            (NC.GraphNodeIDPutProp ToNodeID NodeID
                                                   (for LabelPair in (NC.GraphNodeIDGetProp ToNodeID NodeID)
                                                      collect (OR (CAR LabelPair)
                                                                  LabelPair)))
                                            ToNodeID
                                     else ToNode]

          ;; LAYOUTGRAPH doesn't like duplicate nodes.  These get created when virtual nodes are turned into regular nodes.

                (SETQ Lattice (NC.RemoveDuplicateNodesFromGraph Lattice))
                (NC.RebuildFromNodesInGraph Lattice)
                (AND NewNodes (NC.LayoutNewBrowserNodes NodeToExpand NewNodes BrowserFormat SpecialBrowserSpecs))
                (replace (GRAPH GRAPHNODES) of Graph with Lattice)      ; Build links legend and fix up TONODES in the graph.
                (NC.SetBrowserLinksLegend Card (NC.MakeLinksLegend Graph Window DropVirtualNodesFlg))
                (WINDOWPROP Window 'GRAPH Graph)                        ; Display the graph.
                (NC.ShowBrowserGraph Graph Window)
                (NC.SetSubstance Card Graph)
                (NC.MarkCardDirty Card)
                (NC.ClearMsg Window T])
```

(NC.AskBrowserSpecs

```
[LAMBDA (MainWindow BrowserCard OldLinkLabels OldDepth OldFormat CreatingBrowserFlg Don'tAskFlg)
                                                                ; Edited 21-Oct-88 14:12 by RAR
```

;; Puts up the big stylesheet asking user about link types, depth, browser format, etc. The stylesheet returns a list of 5 things: forward links, ;; backward links, depth, format, and orientation. The last two are smashed together to form a browserformat and the first two are nconc'ed ;; together. Thus we return a list of 3 things: links, depth, and browserformat.

;; rht 4/1/85: Now takes Don'tAskFlg arg for when we don't want to ask the user for browser specs.

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

;; pmi 4/2/87: Added NC.MenuFont to all menus.

;; rht 1/23/88: Now uses OmitLinkLabelsFlg taken off user data props to decide whether link choices appear in stylesheet.

;; RAR 10/21/88 Fixed so that if you can pass ALL or _ALL (and it actually works like the documentation says...)

```
(DECLARE (GLOBALVARS NC.MenuFont MENUFONT NC.BrowserFormatOptions))
(PROG ((LinkLabels (NC.RetrieveLinkLabels (fetch (Card NoteFile) of BrowserCard)
                          T))
       (OmitLinkLabelsFlg (NC.FetchUserDataProp BrowserCard 'OmitLinkLabelsFromBrowserSpecsFlg))
       Position Choices ReverseFlg DepthMenu FormatMenu OrientationMenu DepthSelection FormatSelection
       OrientationSelection)                                    ; Replace ALL and/or _ALL with the actual lists of linklabels and
                                                                ; backlinklabels in oldLinkLabels.
      (SETQ OldLinkLabels (LSUBST LinkLabels 'ALL (LSUBST (NCP.ReverseLinkTypes (fetch (Card NoteFile)
                                                                                       of BrowserCard))
                                                          '_ALL
                                                          OldLinkLabels)))
      [if Don'tAskFlg
          then (RETURN (LIST (OR OldLinkLabels LinkLabels)
                             (OR OldDepth 99999)
```

```
                          (OR OldFormat '(LATTICE]
        [SETQ Position (AND (WINDOWP MainWindow)
                            (create POSITION
                                    XCOORD _ (fetch (REGION LEFT) of (WINDOWPROP MainWindow 'REGION))
                                    YCOORD _ (fetch (REGION TOP) of (WINDOWREGION MainWindow]
        (if CreatingBrowserFlg
            then (SETQ OldLinkLabels LinkLabels))
        (SETQ ReverseFlg (EQ (CADDR OldFormat)
                             'REVERSE))
        (SETQ DepthMenu (create MENU
                                ITEMS _ '(0 1 2 3 4 5 6 7 8 9 INF)
                                MENUFONT _ NC.MenuFont))
        (SETQ DepthSelection (if (OR (NOT (FIXP OldDepth))
                                     (IGREATERP OldDepth 9)
                                     (ILESSP OldDepth 0))
                                 then 'INF
                                 else OldDepth))
        (SETQ FormatMenu (create MENU
                                 ITEMS _ NC.BrowserFormatOptions
                                 MENUFONT _ NC.MenuFont))
        (SETQ FormatSelection (OR (CAR (FASSOC (CAR OldFormat)
                                               NC.BrowserFormatOptions))
                                  'LATTICE))
        (SETQ OrientationMenu (create MENU
                                      ITEMS _ '(Horizontal Vertical Reverse/Horizontal Reverse/Vertical)
                                      MENUFONT _ NC.MenuFont))
        [SETQ OrientationSelection (if (EQ (CADR OldFormat)
                                           'VERTICAL)
                                       then (if ReverseFlg
                                                then 'Reverse/Vertical
                                                else 'Vertical)
                                       else (if ReverseFlg
                                                then 'Reverse/Horizontal
                                                else 'Horizontal]
        [SETQ Choices (COND
                        (OmitLinkLabelsFlg (STYLESHEET (CREATE.STYLE 'ITEMS (LIST DepthMenu FormatMenu
                                                                                  OrientationMenu)
                                                                                  'SELECTIONS
                                                                                  (LIST DepthSelection FormatSelection
                                                                                        OrientationSelection)
                                                                                  'ITEM.TITLES
                                                                                  '(Depth Format Orientation)
                                                                                  'NEED.NOT.FILL.IN
                                                                                  '(NIL NIL NIL)
                                                                                  'POSITION Position 'ITEM.TITLE.FONT
                                                                                  (FONTCOPY MENUFONT 'WEIGHT 'BOLD)
                                                                                  'TITLE "Browser Specs?")))
                        (T (STYLESHEET (CREATE.STYLE 'ITEMS (LIST (create MENU
                                                                          ITEMS _ LinkLabels
                                                                          MENUFONT _ NC.MenuFont)
                                                                  (create MENU
                                                                          ITEMS _
                                                                          (for Link in LinkLabels
                                                                             collect (PACK* '_ Link))
                                                                          MENUFONT _ NC.MenuFont)
                                                                  DepthMenu FormatMenu OrientationMenu)
                                                                  'SELECTIONS
                                                                  (LIST (for Label in OldLinkLabels
                                                                           when (NEQ (NTHCHAR Label 1)
                                                                                     '_)
                                                                           collect Label)
                                                                        (for Label in OldLinkLabels
                                                                           when (EQ (NTHCHAR Label 1)
                                                                                    '_)
                                                                           collect Label)
                                                                        DepthSelection FormatSelection OrientationSelection)
                                                                  'ITEM.TITLES
                                                                  '(Forward% Links Backward% Links Depth Format Orientation)
                                                                  'NEED.NOT.FILL.IN
                                                                  '(MULTI MULTI NIL NIL NIL)
                                                                  'POSITION Position 'ITEM.TITLE.FONT (FONTCOPY MENUFONT
                                                                                                               'WEIGHT
                                                                                                               'BOLD)
                                                                  'TITLE "Browser Specs?"]
        [if (AND OmitLinkLabelsFlg Choices)
            then (SETQ Choices `(,OldLinkLabels NIL ,@Choices]
        (RETURN (COND
                  [Choices (LIST (APPEND (CAR Choices)
                                         (CADR Choices))
                                 (OR (FIXP (CADDR Choices))
                                     MAX.FIXP)
                                 (CONS (CADDDR Choices)
                                       (SELECTQ (CADDDR (CDR Choices))
                                            (Horizontal (LIST 'HORIZONTAL 'REVERSE/DAUGHTERS))
                                            (Vertical (LIST 'VERTICAL NIL))
                                            (Reverse/Horizontal
                                                 (LIST 'HORIZONTAL 'REVERSE 'REVERSE/DAUGHTERS))
```

```
                                        (Reverse/Vertical
                                            (LIST 'VERTICAL 'REVERSE))
                                        NIL]
                    (CreatingBrowserFlg NIL)
                    (T (LIST OldLinkLabels OldDepth OldFormat])
```

(NC.ChangeBrowserSpecs

```
[LAMBDA (Window)                                                    ; Edited 29-Jul-88 20:25 by Trigg
```

;;; Change the values of the various browser specs including link types, browser format, search depth, etc.

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

;; rht 2/7/86: Now sets and gets browser link labels, etc. via fetch/set fns.

;; rht 11/1/86: Added NC.ProtectedCardOperation wrapper and check for ops in progress.

;; rg 3/4/87 rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg

;; rht 7/29/88: No longer calls NC.SetPropListDirtyFlg

```
(LET ((Card (NC.CoerceToCard Window)))
     (NC.ProtectedCardOperation Card "Browser Specs" NIL (PROG (LinkLabels RootNodes BrowserSpecs
                                                                BrowserFormat Depth)
                                                               (SETQ LinkLabels (NC.FetchBrowserLinkLabels
                                                                                 Card))
                                                               (SETQ BrowserFormat (NC.FetchBrowserFormat
                                                                                    Card))
                                                               (SETQ Depth (NC.FetchBrowserDepth Card))
                                                               (SETQ BrowserSpecs
                                                                (NC.AskBrowserSpecs Window Card LinkLabels
                                                                  Depth BrowserFormat))
                                                               (SETQ LinkLabels (CAR BrowserSpecs))
                                                               (SETQ Depth (CADR BrowserSpecs))
                                                               (SETQ BrowserFormat (CADDR BrowserSpecs))
                                                               (NC.SetBrowserLinkLabels Card LinkLabels)
                                                               (NC.SetBrowserFormat Card BrowserFormat)
                                                               (NC.SetBrowserDepth Card Depth)
                                                               (NC.ClearMsg Window T])
```

(NC.AskSpecialBrowserSpecs

```
[LAMBDA (BrowserWindow)                                             (* rht%: "30-May-85 21:44")
```

(* * Get the specification for laying out a browser graph from the user and return them in a BrowserSpecs record.)

(* * rht 3/8/85%: Threw out question about browser Format. That is now obtained in NC.AskBrowserSpecs along with link types, etc.)

```
   (PROG (BrowserSpecs)
         (OR (WINDOWP BrowserWindow)
             (SETQ BrowserWindow))
         [SETQ BrowserSpecs (create SPECIALBROWSERSPECS
                                    Font _ NIL
                                    MotherD _ (FIXP (MKATOM (NC.AskUser "What is the MotherD for this browser? "
                                                                        NIL NIL T BrowserWindow T)))
                                    PersonalD _ (FIXP (MKATOM (OR (NC.AskUser "What is the PersoalD for this
                                                                               browser? " NIL NIL T BrowserWindow T)
                                                                  10)))
                                    FamilyD _ (FIXP (MKATOM (NC.AskUser "What is the FamilyD for this browser? "
                                                                        NIL NIL T BrowserWindow T]
         (RETURN BrowserSpecs])
```

(NC.BrowserFlipRoots

```
[LAMBDA (Window GraphCard GraphNodes RootCards)                     (* rht%: "18-Nov-85 21:08")
```

(* * Flip to reverse video for each root node in the graph.)

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

```
   (for RootCard in RootCards bind RootNode do (SETQ RootNode (FASSOC (NC.GetBrowserNodeID GraphCard RootCard)
                                                                      GraphNodes))
                                               (AND RootNode (FLIPNODE RootNode Window])
```

(NC.ChangeBrowserRoots

```
[LAMBDA (Window)                                                    ; Edited 29-Jul-88 20:12 by Trigg
```

;; Change which nodes in the browser are considered roots, e.g. for purposes of recompute.

```
(LET ((Card (NC.CoerceToCard Window)))
     (NC.ProtectedCardOperation Card "Respecify browser roots" NIL
            (LET (NewRootCards)
                 (SETQ NewRootCards (NC.RespecifyBrowserRoots Card (NC.FetchBrowserRoots Card)
                                                              (fetch (GRAPH GRAPHNODES) of (WINDOWPROP Window 'GRAPH))
                                                              Window))
                 (if (LISTP NewRootCards)
                     then (NC.SetBrowserRoots Card NewRootCards)
                          (NC.MarkCardDirty Card))
```

```
                    NewRootCards])
```

(NC.RespecifyBrowserRoots

```
[LAMBDA (BrowserCard CurrentRootCards GraphNodes Window)       ; Edited 29-Jul-88 20:31 by Trigg
```

;; Let user select a new set of root nodes in the browser.

```
(DECLARE (GLOBALVARS NC.SelectingBrowserSourceMenu))
(NC.BrowserFlipRoots Window BrowserCard GraphNodes CurrentRootCards)
(PROG1 (NC.SelectNoteCards NIL NIL NC.SelectingBrowserSourceMenu Window (CONCAT "Please shift-select the
                                                                                Cards and/or Boxes the
                                                                                browser should start from."
                                                                                (CHARACTER 13)
                                                                                "(Current roots are
                                                                                highlighted.)")
                           T)
       (NC.BrowserFlipRoots Window BrowserCard GraphNodes CurrentRootCards])
```

(NC.RebuildFromNodesInGraph

```
[LAMBDA (GraphNodes)                                           (* rht%: "31-Aug-85 16:50")
```

(* * Remove the FROMNODES from every graph node and rebuild them using the TONODES.
Note that there must not be link param thingies in the TONODES, but virtual nodes are okay.)

```
(for Node in GraphNodes bind NodeID NewFromNodes eachtime (BLOCK)
   do (SETQ NodeID (fetch (GRAPHNODE NODEID) of Node))
      (SETQ NewFromNodes (for OtherNode in GraphNodes when (FMEMB NodeID (fetch (GRAPHNODE TONODES)
                                                                            of OtherNode))
                            collect (fetch (GRAPHNODE NODEID) of OtherNode)))
      (replace (GRAPHNODE FROMNODES) of Node with (INTERSECTION NewFromNodes NewFromNodes])
```

(NC.RemoveDuplicateNodesFromGraph

```
[LAMBDA (GraphNodes)                                           (* rht%: " 6-Jul-86 17:28")
```

(* * There should be no virtual nodes or link param thingies. This removes duplicate nodes coalescing their TONODES.)

```
(LET (DeletedNodeIDs)
     (SETQ GraphNodes (for Node in GraphNodes bind NodeID AlreadyVisitedFlg
                         eachtime (BLOCK)
                                  (SETQ NodeID (NC.CoerceToGraphNodeIDOrLabel Node))
                                  (if (AND (SETQ AlreadyVisitedFlg (NC.GraphNodeIDGetProp NodeID
                                                                                          'AlreadyVisitedFlg))
                                           (NOT (FMEMB NodeID DeletedNodeIDs)))
                                      then (push DeletedNodeIDs NodeID))
                                  (NC.GraphNodeIDPutProp NodeID 'CumulativeToNodesList
                                         (UNION (NC.GraphNodeIDGetProp NodeID 'CumulativeToNodesList)
                                                (fetch (GRAPHNODE TONODES) of Node)))
                         unless AlreadyVisitedFlg collect (NC.GraphNodeIDPutProp NodeID 'AlreadyVisitedFlg T)
                                                          Node))
     [for NodeID in DeletedNodeIDs bind GraphNode do (SETQ GraphNode (FASSOC NodeID GraphNodes))
                                                     (replace (GRAPHNODE TONODES) of GraphNode
                                                        with (NC.GraphNodeIDGetProp NodeID
                                                                                    'CumulativeToNodesList]
     (for Node in GraphNodes bind NodeID do (NC.GraphNodeIDRemProp (SETQ NodeID (NC.CoerceToGraphNodeIDOrLabel
                                                                                  Node))
                                                                   'CumulativeToNodesList)
                                            (NC.GraphNodeIDRemProp NodeID 'AlreadyVisitedFlg))
     GraphNodes])
```

(NC.ShowBrowserGraph

```
[LAMBDA (Graph Window)                                         ; Edited 29-Sep-88 10:59 by jrc
```

(* * SHOWGRAPH Graph in Window.)

(* * rht 5/13/87%: Added call to NC.InstallCopyButtonEventFn.)

(* * rht 10/13/87%: Added COPYBUTTONEVENTFN arg to SHOWGRAPH to fix bug reported by John Tang.)

(* * jrc 28-sep-88 Saves various WINDOWPROPs that SHOWGRAPH seems to smash.
The original code made assumptions about what the value of each of the WINDOWPROPs was.)

```
(LET [(OldRepaintFn (WINDOWPROP Window 'REPAINTFN))
      (OldRightButtonFn (WINDOWPROP Window 'RIGHTBUTTONFN))
      (OldScrollFn (WINDOWPROP Window 'SCROLLFN))
      (OldReshapeFn (WINDOWPROP Window 'RESHAPEFN]
     (SHOWGRAPH Graph Window (FUNCTION NC.GraphCardLeftButtonFn)
            (FUNCTION NC.GraphCardMiddleButtonFn)
            NIL T (FUNCTION NC.GRAPHERCOPYBUTTONEVENTFN))

     (* * Have to reset windowprops since SHOWGRAPH messes with them.)
                                                    (* Disable the old-style right button grapher editor menu.)
     (WINDOWPROP Window 'RIGHTBUTTONFN OldRightButtonFn)
     (WINDOWPROP Window 'REPAINTFN OldRepaintFn)
     (WINDOWPROP Window 'SCROLLFN OldScrollFn)
```

```
        (WINDOWPROP Window 'RESHAPEFN OldReshapeFn)
        (NC.InstallTitleBarButtonEventFn Window (FUNCTION NC.TitleBarButtonEventFn))
        (NC.InstallCopyButtonEventFn Window])

)
```

;;; Graph editor menu functions.

;;; These moved to GRAPHCARD since many are used by both.

(* * FNS NC.SetUpGraphEditMenus NC.GetGraphEditMenu NC.BrowserRightButtonFn NC.BrowserCreateCardFn NC.BrowserAddLabelFn NC.BrowserChangeLabelFn NC.BrowserAddNodeFn NC.BrowserCreateLinkFn NC.BrowserCreateGlobalLinkFn NC.BrowserAddLink NC.BrowserAddGlobalLink NC.BrowserAddEdgeFn NC.BrowserDeleteCardFn NC.BrowserRemoveNodeFn NC.BrowserDeleteLinkFn NC.BrowserRemoveEdgeFn NC.BrowserShrinkLabelFn NC.BrowserGrowLabelFn NC.BrowserMoveNodeFn NC.BrowserToggleShadeFn NC.CursorInsideGraphNodeP NC.BrowserMoveNodesInRegionFn NC.BrowserMoveSubtreeFn NC.BrowserFixGraphEditMenuFn NC.BrowserCreateCard NC.BrowserCreateLink NC.BrowserDeleteLink NC.BrowserAddNode NC.BrowserAddLabel NC.BrowserAddEdge NC.BrowserRemoveNode NC.DelBrowserContentsLink NC.BrowserRemoveEdge NC.BrowserChangeLabel NC.BrowserGetConfirmation)

;;; Grapher hacks for browser

```
(DEFINEQ
```

(NC.MakeLinksLegend

```
  [LAMBDA (Graph Win DropVirtualNodesFlg)                    (* rht%: " 6-Jul-86 17:28")
```

(* * For every node in the lattice, there should be properties off of its NODEID for each node it's connected to. The values of these props are lists of linklabels. Change these values to also contain the dashing number by assigning a unique dashing number to each new label we come across. If the global var NC.LinkDashingInBrowser is non-nil, then put out a menu serving as a legend mapping link label names to dashing styles. If not, then the menu just contains names of link labels.)

(* * rht 3/9/85%: Modified to use Danny's grapher improvements. Now changes destination nodes to be in the new list format.)

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

```
    (PROG (LabelPairs (MaxDashingStylesNum (LENGTH NC.DashingStyles))
                  ReferencedNodes NumAppearances OldNumAppearances UnderlyingNodeID)
          [for Node in (fetch (GRAPH GRAPHNODES) of Graph) bind NodeID (LabelNum _ 0) eachtime (BLOCK)
             do
             (if DropVirtualNodesFlg
                 then                                        (* Throw away the border indicating a virtual node.)
                      (replace (GRAPHNODE NODEBORDER) of Node with NIL))
             (SETQ NodeID (fetch (GRAPHNODE NODEID) of Node))
             (NC.GraphNodeIDPutProp (SETQ UnderlyingNodeID (OR (NC.CoerceToGraphNodeID Node)
                                                               NodeID))
                    'NumAppearances
                    (if (SETQ OldNumAppearances (NC.GraphNodeIDGetProp UnderlyingNodeID 'NumAppearances))
                        then (ADD1 OldNumAppearances)
                      else 1))
             (if (NC.LinkIconImageObjP (fetch (GRAPHNODE NODELABEL) of Node))
                 then
                 (replace (GRAPHNODE TONODES) of Node
                    with
                    (for DestNode in (fetch (GRAPHNODE TONODES) of Node) eachtime (BLOCK)
                       bind NewLabelPairs Labels DestNodeID NewDestNode NotLabelNodeFlg
                       join                                  (* If already computed a LinkParams list, then rip out the ID.)
                       (if (EQ (CAR DestNode)
                               LINKPARAMS)
                           then (SETQ DestNode (CADR DestNode))) (* Check for virtual nodes.)
                       (SETQ DestNodeID (if (LISTP DestNode)
                                            then (CAR DestNode)
                                          else DestNode))
                       (SETQ NewDestNode (if DropVirtualNodesFlg
                                             then DestNodeID
                                           else DestNode))
                       (SETQ NotLabelNodeFlg (NC.CardFromBrowserNodeID DestNodeID))
                                                             (* Turn forward labels into pairs by adding dashing numbers.)
                       [SETQ NewLabelPairs
                        (if [AND NotLabelNodeFlg (NOT (OR (LISTP (CAR (NC.GraphNodeIDGetProp NodeID DestNodeID)))
                                                          (LISTP (CAR (NC.GraphNodeIDGetProp DestNodeID NodeID]
                            then                             (* Okay to continue since we haven't visited this pair already.)
                                 (APPEND (if (SETQ Labels (NC.GraphNodeIDGetProp NodeID DestNodeID))
                                             then (NC.GraphNodeIDPutProp
                                                    NodeID DestNodeID
                                                    (for Label in Labels bind Pair
                                                       collect [COND
                                                                 ((NULL (SETQ Pair (FASSOC Label LabelPairs)))
                                                                  [SETQ Pair (CONS Label (COND
                                                                                           ((ILESSP LabelNum
                                                                                               MaxDashingStylesNum
                                                                                               )
                                                                                            (SETQ LabelNum
                                                                                             (ADD1 LabelNum)))
```

```
                                                                    (T LabelNum]
                                                          (SETQ LabelPairs (CONS Pair LabelPairs]
                                                       Pair)))
                                              (if (SETQ Labels (NC.GraphNodeIDGetProp DestNodeID NodeID))
                                                  then (NC.GraphNodeIDPutProp
                                                        DestNodeID NodeID
                                                        (for Label in Labels bind Pair
                                                           collect [COND
                                                                      ((NULL (SETQ Pair (FASSOC Label LabelPairs)))
                                                                       [SETQ Pair (CONS Label (COND
                                                                                               ((ILESSP LabelNum
                                                                                                    MaxDashingStylesNum
                                                                                                       )
                                                                                                (SETQ LabelNum
                                                                                                 (ADD1 LabelNum)))
                                                                                               (T LabelNum]
                                                          (SETQ LabelPairs (CONS Pair LabelPairs]
                                                    Pair]  (* Likewise for backward labels.)
                    (if NewLabelPairs
                        then

(* Stick this dest node on the referenced list since we know a node points to it.)

                                 (if (NOT (FMEMB NewDestNode ReferencedNodes))
                                     then (push ReferencedNodes NewDestNode))
                                 [LIST (COND
                                          ((CDR NewLabelPairs)

(* There are multiple links joining these two nodes so record nodeids in param list so we can draw flower of links.)

                                           (LIST LINKPARAMS NewDestNode 'DRAWLINKFN (FUNCTION
                                                                                       NC.BrowserDrawLinkFn)
                                                 'NODEID NodeID 'DESTNODEID DestNodeID))
                                          (T                        (* Only one link, so compute dashing style here.)
                                                                    (* Check whether link is forward or backward and throw in
                                                                    backward flag in appropriate.)
                                             (if (NC.GraphNodeIDGetProp NodeID DestNodeID)
                                                 then [LIST LINKPARAMS NewDestNode 'DRAWLINKFN
                                                            (FUNCTION NC.BrowserDrawLinkFn)
                                                            'DASHING
                                                            (CAR (FNTH NC.DashingStyles (CDAR NewLabelPairs]
                                               else (LIST LINKPARAMS NewDestNode 'DRAWLINKFN
                                                          (FUNCTION NC.BrowserDrawLinkFn)
                                                          'DASHING
                                                          (CAR (FNTH NC.DashingStyles (CDAR NewLabelPairs)))
                                                          'BACKWARDFLG T]
                    else

(* Stick this dest node on the referenced list since we know a node points to it.)

                                 (if (NOT (FMEMB DestNodeID ReferencedNodes))
                                     then (push ReferencedNodes DestNodeID))
                                 (if (NOT NotLabelNodeFlg)
                                     then (LIST DestNodeID)
                                   else NIL]

(* * Note that the following loop gains time at the expense of space.
The space-efficient version would only generate cons nodes for nodes to be deleted, but would require in general, several
walks through the structure.)

(* Delete all nodes except the ones that either point to something or are pointed to.
But keep those unreferenced nodes that appear exactly once in the graph.
They'll wind up being roots.)

(replace (GRAPH GRAPHNODES) of Graph
   with (for Node in (fetch (GRAPH GRAPHNODES) of Graph) eachtime (BLOCK)
           when (LET* [(UnderlyingNodeID (OR (NC.CoerceToGraphNodeID Node)
                                             (fetch (GRAPHNODE NODEID) of Node)))
                       (NumAppearances (NC.GraphNodeIDGetProp UnderlyingNodeID 'NumAppearances]
                      (if (OR (fetch (GRAPHNODE TONODES) of Node)
                              (FMEMB (fetch (GRAPHNODE NODEID) of Node)
                                     ReferencedNodes)
                              (EQ NumAppearances 1))
                        else                                (* This node is getting deleted.)
                             (NC.GraphNodeIDPutProp UnderlyingNodeID 'NumAppearances (SUB1
                                                                                        NumAppearances
                                                                                        ))
                             NIL))
           collect Node))

(* Get rid of node borders for virtual nodes that now only appear once in the graph.
Also clean off prop list.)

[for Node in (fetch (GRAPH GRAPHNODES) of Graph)
   do (LET [(UnderlyingNodeID (OR (NC.CoerceToGraphNodeID Node)
                                  (fetch (GRAPHNODE NODEID) of Node]
           (if (EQ 1 (NC.GraphNodeIDGetProp UnderlyingNodeID 'NumAppearances))
```

```
                    then (replace (GRAPHNODE NODEBORDER) of Node with NIL))
                (NC.GraphNodeIDRemProp UnderlyingNodeID 'NumAppearances]
        (SETQ LabelPairs (DREVERSE LabelPairs))
        (AND Win (NC.MakeLinksLegendMenu Win LabelPairs))
        (RETURN LabelPairs])
```

(NC.MakeLinksLegendMenu

```
[LAMBDA (Win LabelPairs)                                    ; Edited 23-Jan-89 13:52 by rtk
```

;; Build a links legend menu and attach to Win

;; rht 1/10/85: Before starting, kill any old links legend menus for Win.

;; rht 1/13/86: Now holds onto value of PASSTOMAINCOMS windowprop of prompt win and restores after reattaching.

;; rht 1/15/86: Added windowprops MINSIZE and MAXSIZE to fix the bug where reshaping browser screws up links legend menu.

;; rht 3/7/86: Now closes prompt window before attaching menu. Uses ATTACHMENU to attach the menu.

;; rht 4/5/86: Took out call to NC.MoveWindowOntoScreen. For big browsers it causes redraw of window which is too high a price to pay.

;; rht 3/20/87: Changed so that ATTACHMENU call is inside of NC.WithWindowsUnattached macro. Also took out closing of prompt window, as it's no longer necessary.

;; rht 1/16/88: Now does nothing if card has non-nil OmitLinksLegendFlg user data prop.

;; pmi 8/18/88: No longer puts up a link legend if the browser does not have any links.

;; bk 1/11/89: Add check for the fixed graph card through FIXED-GRAPH-MENU-WINDOW prop.

;; bk 1/23/89: Remove prompt window before adding attached windows. Also close the fixed menu window & re-created it later.

```
(DECLARE (GLOBALVARS NC.LinkDashingInBrowser))
(OR (NCP.CardUserDataProp (NCP.WhichCard Win)
          'OmitLinksLegendFlg)
    (LET ((HAS-GRAPH-MENU (WINDOWPROP Win 'FIXED-GRAPH-MENU-WINDOW))
          Menu MenuWin PromptWin MainWinPromptInfo PromptWinPASSTOMAINCOMS)
         (REMOVEPROMPTWINDOW Win)
         (AND HAS-GRAPH-MENU (CLOSEW HAS-GRAPH-MENU))
         (for AttachedWin in (ATTACHEDWINDOWS Win) when (WINDOWPROP AttachedWin 'LINKSLEGENDWINP)
            do (CLOSEW AttachedWin))
         (REPOSITIONATTACHEDWINDOWS Window)
         (if LabelPairs
             then [SETQ Menu (COND
                                (NC.LinkDashingInBrowser (create MENU
                                                                 ITEMS _
                                                                 [for Pair in LabelPairs
                                                                    join (LIST (CAR Pair)
                                                                               (LIST '"  "]
                                                                 TITLE _ 'Links
                                                                 MENUCOLUMNS _ 2))
                                (T (create MENU
                                           ITEMS _ (for Pair in LabelPairs collect (CAR Pair))
                                           TITLE _ 'Links
                                           MENUCOLUMNS _ 1]    ; Stick the links legend window at upper right corner.
                  [NC.WithTopWindowsUnattached Win (SETQ MenuWin (ATTACHMENU Menu Win 'RIGHT 'TOP]
                                                              ; Rig so that close of menu won't close browser.
                  (WINDOWDELPROP MenuWin 'PASSTOMAINCOMS 'CLOSEW)
                  (WINDOWADDPROP MenuWin 'CLOSEFN [FUNCTION (LAMBDA (W)
                                                            (DETACHWINDOW W]
                         T)
                  (WINDOWADDPROP MenuWin 'REPAINTFN 'NC.LinksLegendRepaintFn)
                  (WINDOWADDPROP MenuWin 'RESHAPEFN 'NC.LinksLegendReshapeFn)
                  (WINDOWADDPROP MenuWin 'LINKSLEGENDWINP T)
                  (WINDOWPROP Win 'NCLABELPAIRS LabelPairs)
                  (if NC.LinkDashingInBrowser
                      then (NC.LinksLegendRepaintFn MenuWin NIL)))
         (if HAS-GRAPH-MENU
             then (NC.BrowserFixGraphEditMenuFn Win])
```

(NC.LinksLegendRepaintFn

```
[LAMBDA (Win Region)                                        (* rht%: "23-Sep-85 15:47")
```

(* * Repaint the right-hand column dashing in the browser's links legend menu.)

(* * rht 7/15/85%: Added bail out for case of screwy 1.1 files.)

```
  (PROG [[Menu (CAR (WINDOWPROP Win 'MENU]
         Items
         (LabelPairs (WINDOWPROP (MAINWINDOW Win)
                                 'NCLABELPAIRS]
        (SETQ Items (fetch (MENU ITEMS) of Menu))
```

(* This little bail out is for case of 1.1 files where label pairs are screwed up and so are menu items.)

```
        (if (NULL (CAR Items))
            then (RETURN NIL))
        (for Item in (CDR Items) by (CDDR Item) as LabelPair in LabelPairs bind ItemRegion
           do (SETQ ItemRegion (MENUITEMREGION Item Menu))
              (DRAWLINE (IPLUS 4 (fetch (REGION LEFT) of ItemRegion))
                        (IPLUS (QUOTIENT (fetch (REGION HEIGHT) of ItemRegion)
```

```
                                2)
                        (fetch (REGION BOTTOM) of ItemRegion))
                (IDIFFERENCE (IPLUS (fetch (REGION WIDTH) of ItemRegion)
                                    (fetch (REGION LEFT) of ItemRegion))
                        4)
                (IPLUS (QUOTIENT (fetch (REGION HEIGHT) of ItemRegion)
                                2)
                        (fetch (REGION BOTTOM) of ItemRegion))
                1 NIL Win NIL (CAR (FNTH NC.DashingStyles (CDR LabelPair])
```

(NC.BrowserDrawLinkFn

```
[LAMBDA (X1 Y1 X2 Y2 Width Operation Win Color Dashing ParamList)
                                                                    ; Edited 27-Jul-90 09:21 by tafel
```

;;; This is called by grapher to draw a link. If there's no NODEID param then just one link to draw. Check the dashing global var and draw it. Otherwise, ;;; there are multiple links so we need to draw a flower arrangement. Note the ugliness that NC.LinkDashingInBrowser being off means that flowers ;;; won't get drawn. If the arrow heads global var is on, then draw arrow head at one end.

```
    (PROG (NodeID DestNodeID MidpointFlg ClippingRegion)
          (COND
             [NC.LinkDashingInBrowser (COND
                                         ((SETQ NodeID (LISTGET ParamList 'NODEID))
                                                                    ; Multiple links. Have to draw a flower of spline curves for the
                                                                    ; links.
                                          (RETURN (NC.DrawFlowerLinks NodeID (SETQ DestNodeID (LISTGET
                                                                                               ParamList
                                                                                               'DESTNODEID))
                                                                                              X1 Y1 X2 Y2 Width Operation Win Color]
             (T (SETQ Dashing NIL)))
          (COND
             ([LET ((ClippingRegion (DSPCLIPPINGREGION NIL Win)))
                   ;; determines if the region in which the line will be drawn intersects the window's clipping region (increased by the arrow head
                   ;; length). We do not use REGIONSINTERSECTP because we would have to cons up the regions just to have them pulled
                   ;; apart.
                   (OR (IGREATERP (MIN X1 X2)
                                  (PLUS NC.ArrowHeadLength (fetch (REGION RIGHT) of ClippingRegion)))
                       (IGREATERP (DIFFERENCE (fetch (REGION LEFT) of ClippingRegion)
                                              NC.ArrowHeadLength)
                                  (IMAX X1 X2))
                       (IGREATERP (MIN Y1 Y2)
                                  (PLUS NC.ArrowHeadLength (fetch (REGION TOP) of ClippingRegion)))
                       (IGREATERP (DIFFERENCE (fetch (REGION BOTTOM) of ClippingRegion)
                                              NC.ArrowHeadLength)
                                  (IMAX Y1 Y2]
              (RETURN)))
          (DRAWLINE X1 Y1 X2 Y2 Width Operation Win Color Dashing)
          (COND
             ((NEQ NC.ArrowHeadsInBrowser 'None)
              (SETQ MidpointFlg (EQ NC.ArrowHeadsInBrowser 'AtMidpoint))
              (COND
                 ((LISTGET ParamList 'BACKWARDFLG)
                  (NC.DrawArrowHead X2 Y2 X1 Y1 Width Operation Win Color MidpointFlg))
                 (T (NC.DrawArrowHead X1 Y1 X2 Y2 Width Operation Win Color MidpointFlg])
```

(NC.DrawFlowerLinks

```
[LAMBDA (NodeID1 NodeID2 X1 Y1 X2 Y2 Width Operation Stream Color)
                                                                    (* rht%: " 6-Jul-86 17:27")
```

(* * Expects to find a list of pairs on Node1's ID's proplist under the property with name Node2's ID
(or vice versa) These are pairs of label and dashing number. For each one, draw a spline with one knot using given dashing number. The more we draw, the farther each gets from the center line.
The very first is along the center line. Subsequent splines alternate on either side of the center line.)

(* * rht 3/9/85%: Now draws first the forward links and then the backward links.)

```
    (PROG ((Count -1))
          (for Pair in (NC.GraphNodeIDGetProp NodeID1 NodeID2) do (NC.DrawFlowerLink X1 Y1 X2 Y2
                                                                                     (LIST 'ROUND Width Color)
                                                                                     (CAR (FNTH NC.DashingStyles (CDR Pair)))
                                                                                     (SETQ Count (ADD1 Count))
                                                                                     Stream Width Operation Color))
          (for Pair in (NC.GraphNodeIDGetProp NodeID2 NodeID1) do (NC.DrawFlowerLink X2 Y2 X1 Y1
                                                                                     (LIST 'ROUND Width Color)
                                                                                     (CAR (FNTH NC.DashingStyles (CDR Pair)))
                                                                                     (SETQ Count (ADD1 Count))
                                                                                     Stream Width Operation Color])
```

(NC.DrawFlowerLink

```
[LAMBDA (X1 Y1 X2 Y2 Brush Dashing Num Win Width Operation Color)
                                                                    (* rht%: " 1-May-87 14:55")
```

(* * Draw one link between given points according to given Dashing.
If Num is 0, then draw straight line. Otherwise, draw curves above for odd Num and below for even Num, getting wider for bigger values of Num.)

(* * rht 3/21/86%: Now checks for WINDOWPROP of Scale for when we're in a browser overview window.)

(* * rht 5/1/87%: Changed bogus mention of "Stream" to "Win.")

```
(DECLARE (GLOBALVARS NC.ArrowHeadsInBrowser NC.GraphFlowerLinkSeparation))
(PROG ((MidpointX (QUOTIENT (IPLUS X1 X2)
                            2))
       (MidpointY (QUOTIENT (IPLUS Y1 Y2)
                            2))
       XOffset YOffset Scale Window WindowScale)
      (COND
         ((ZEROP Num)
          (DRAWLINE X1 Y1 X2 Y2 Width Operation Win Color Dashing)
          (AND (NEQ NC.ArrowHeadsInBrowser 'None)
               (NC.DrawArrowHead X1 Y1 X2 Y2 Width Operation Win Color T)))
         (T
```

(* Check slope of line. If < 45 degrees, then make spline nodes be vertically removed, otherwise horizontally removed.)

```
            (COND
               ((GREATERP (ABS (DIFFERENCE X2 X1))
                          (ABS (DIFFERENCE Y2 Y1)))
                [SETQ YOffset (COND
                                 ((ODDP Num)
                                  (TIMES NC.GraphFlowerLinkSeparation (QUOTIENT (ADD1 Num)
                                                                                2)))
                                 (T (MINUS (TIMES NC.GraphFlowerLinkSeparation (QUOTIENT (ADD1 Num)
                                                                                         2]
                (SETQ XOffset 0))
               (T [SETQ XOffset (COND
                                   ((ODDP Num)
                                    (TIMES NC.GraphFlowerLinkSeparation (QUOTIENT (ADD1 Num)
                                                                                  2)))
                                   (T (MINUS (TIMES NC.GraphFlowerLinkSeparation (QUOTIENT (ADD1 Num)
                                                                                           2]
                  (SETQ YOffset 0)))
            (SETQ Scale (DSPSCALE NIL Win))
            [SETQ Window (AND (DISPLAYSTREAMP Win)
                              (WINDOWP (WFROMDS Win]
            (if [AND Window (SETQ WindowScale (WINDOWPROP Window 'Scale]
                then (SETQ Scale (FTIMES Scale WindowScale)))
            (SETQ XOffset (FIXR (TIMES XOffset Scale)))
            (SETQ YOffset (FIXR (TIMES YOffset Scale)))
            (DRAWLINE X1 Y1 (PLUS MidpointX XOffset)
                   (PLUS MidpointY YOffset)
                   Width Operation Win NIL Dashing)
            (DRAWLINE (PLUS MidpointX XOffset)
                   (PLUS MidpointY YOffset)
                   X2 Y2 Width Operation Win NIL Dashing)
            (AND (NEQ NC.ArrowHeadsInBrowser 'None)
                 (NC.DrawArrowHead X1 Y1 (PLUS MidpointX XOffset)
                        (PLUS MidpointY YOffset)
                        Width Operation Win Color NIL])
```

(NC.LinksLegendReshapeFn

```
[LAMBDA (Window OldImage OldRegion)                    (* rht%: " 7-Dec-84 12:15")
```

(* * Called when main window is reshaped. Just redisplays the links legend.)

```
   (REDISPLAYW Window])
```

(NC.DrawArrowHead

```
[LAMBDA (X1 Y1 X2 Y2 Width Operation Stream Color MidpointFlg XOffset YOffset)
                                                       (* rht%: "26-Mar-86 15:45")
```

(* * Draw an arrowhead on the end of the line segment from X1,Y1 to X2,Y2.)

(* * rht 8/12/85%: Added check for whether point is inside clipping region before computing arrow head.)

(* * rht 2/28/86%: Now checks for Scale WINDOWPROP on the window.
If so, then multiply that in with Scale. We're probably in a shrunken browser in that case.)

```
   (PROG ((XDiff (FLOAT (IDIFFERENCE X1 X2)))
          (YDiff (FLOAT (IDIFFERENCE Y1 Y2)))
          [Window (AND (DISPLAYSTREAMP Stream)
                       (WINDOWP (WFROMDS Stream]
          (Scale (DSPSCALE NIL Stream))
          WindowScale LineLength A11 A12 A21 A22 Tmp1 Tmp2 ClippingRegion ArrowHeadXVal ArrowHeadYVal
          ArrowHeadLength)
         (OR XOffset (SETQ XOffset 0))
         (OR YOffset (SETQ YOffset 0))
         (if [AND Window (SETQ WindowScale (WINDOWPROP Window 'Scale]
             then (SETQ Scale (FTIMES Scale WindowScale)))
         (SETQ ArrowHeadXVal (FIXR (TIMES NC.ArrowHeadXVal Scale)))
```

```
          (SETQ ArrowHeadYVal (FIXR (TIMES NC.ArrowHeadYVal Scale)))
          (SETQ ArrowHeadLength (FIXR (TIMES NC.ArrowHeadLength Scale)))
          [COND
             (MidpointFlg (SETQ XOffset (PLUS XOffset (DIFFERENCE (QUOTIENT (IPLUS X1 X2)
                                                                            2)
                                                                  X2)))
                          (SETQ YOffset (PLUS YOffset (DIFFERENCE (QUOTIENT (IPLUS Y1 Y2)
                                                                            2)
                                                                  Y2]
```

(* If the point at which the arrow head will be drawn isn't inside the clipping region together with a border, then bail out.)

```
          (SETQ ClippingRegion (DSPCLIPPINGREGION NIL Stream))
          (COND
             ([AND ClippingRegion (NOT (INSIDEP (CREATEREGION (DIFFERENCE (fetch (REGION LEFT) of ClippingRegion)
                                                                          ArrowHeadLength)
                                                              (DIFFERENCE (fetch (REGION BOTTOM) of ClippingRegion)
                                                                     ArrowHeadLength)
                                                              (PLUS (fetch (REGION WIDTH) of ClippingRegion)
                                                                    ArrowHeadLength ArrowHeadLength)
                                                              (PLUS (fetch (REGION HEIGHT) of ClippingRegion)
                                                                    ArrowHeadLength ArrowHeadLength))
                                                (PLUS X2 XOffset)
                                                (PLUS Y2 YOffset]
              (RETURN)))
          [SETQ LineLength (SQRT (PLUS (TIMES XDiff XDiff)
                                       (TIMES YDiff YDiff]
          (SETQ A11 (QUOTIENT (FLOAT XDiff)
                              LineLength))
          (SETQ A21 (QUOTIENT (FLOAT YDiff)
                              LineLength))
          (SETQ A12 (MINUS A21))
          (SETQ A22 A11)
          (SETQ Tmp1 (TIMES A11 ArrowHeadXVal))
          (SETQ Tmp2 (TIMES A21 ArrowHeadXVal))
          (SETQ X2 (PLUS X2 XOffset))
          (SETQ Y2 (PLUS Y2 YOffset))
          (DRAWLINE X2 Y2 (PLUS X2 Tmp1 (TIMES A12 ArrowHeadYVal))
                 (PLUS Y2 Tmp2 (TIMES A22 ArrowHeadYVal))
                 Width Operation Stream Color)
          (DRAWLINE X2 Y2 (PLUS X2 Tmp1 (TIMES A12 (MINUS ArrowHeadYVal)))
                 (PLUS Y2 Tmp2 (TIMES A22 (MINUS ArrowHeadYVal)))
                 Width Operation Stream Color])

)
```

;;; for making and manipulating the tiny attached shrunken browser window.

```
(RPAQ? NC.BrowserOverviewDefaultWidth 75)

(RPAQ? NC.BrowserOverviewDefaultHeight 75)

(RPAQ? NC.LeastScaleForGraphNodeShrinking 0.3)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.BrowserOverviewDefaultWidth NC.BrowserOverviewDefaultHeight NC.LeastScaleForGraphNodeShrinking
       NC.BrowserOverviewSpecsStylesheet NC.DefaultWhereToAttachOverviewWin NC.DefaultBrowserOverviewMode
       NC.OverviewWinMode.Compress NC.OverviewWinMode.Expand)
)

(RPAQ? NC.OverviewWinMode.Compress '|Compress Overview Win|)

(RPAQ? NC.OverviewWinMode.Expand 'Expand% Overview)

(RPAQ? NC.DefaultBrowserOverviewMode 'Neither)

(RPAQ? NC.DefaultWhereToAttachOverviewWin '(LEFT . TOP))

(RPAQ? NC.BrowserOverviewSpecsStylesheet
       [CREATE.STYLE 'TITLE "Choose browser overview specs" 'ITEM.TITLES '(Edge |Position on Edge| Mode)
              'ITEM.TITLE.FONT
              (FONTCOPY MENUFONT 'WEIGHT 'BOLD)
              'ITEMS
              (LIST [create MENU ITEMS _ '((LEFT LEFT "Position along left edge.")
                                           (TOP TOP "Position along top edge.")
                                           (BOTTOM BOTTOM "Position along bottom edge."]
                    [create MENU ITEMS _ '((TOP/RIGHT TOP "Position at top or right end of edge.")
                                           (CENTER CENTER "Position at center of edge.")
                                           (BOTTOM/LEFT BOTTOM "Position at bottom or left of edge."]
                    (create MENU ITEMS _ '((|Compress Overview Win| |Compress Overview Win| "Compress the
                                                  overview window to exactly fit the overview contents.")
                                           (Expand% Overview Expand% Overview "Expand the overview contents to
                                                  exactly fill the overview window.")
                                           (Neither Neither "Neither expand the overview contents nor compress
                                                  the overview window."])
```

```
(DEFINEQ

(NC.MakeBrowserOverviewWin
  [LAMBDA (BrowserWin Region)                                    (* Randy.Gobbel " 4-Mar-87 13:57")

          (* * Make and attach a little window to contain a shrunken bitmap of the entire browser.
          If Region arg is passed then use that as size for overview win, else use defaults.)

          (* * rht 3/7/86%: Now uses stylesheet to figure out where to attach.)

          (* * rht 11/1/86%: Added NC.ProtectedCardOperation wrapper and check for ops in progress.)

          (* * rg |3/4/87| rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg)

    (LET ((Card (NC.CoerceToCard BrowserWin)))
         (NC.ProtectedCardOperation Card "Build Browser Overview" NIL
                (LET ((OverviewWinWidth (if Region
                                            then (fetch (REGION WIDTH) of Region)
                                          else (OR (WINDOWPROP BrowserWin 'OverviewWinWidth)
                                                   NC.BrowserOverviewDefaultWidth)))
                      (OverviewWinHeight (if Region
                                             then (fetch (REGION HEIGHT) of Region)
                                           else (OR (WINDOWPROP BrowserWin 'OverviewWinHeight)
                                                    NC.BrowserOverviewDefaultHeight)))
                      [OverviewWin (OPENWP (WINDOWPROP BrowserWin 'BrowserOverviewWin]
                      (WhereToAttach (OR (WINDOWPROP BrowserWin 'WHERETOATTACHOVERVIEWWIN)
                                         NC.DefaultWhereToAttachOverviewWin)))
                     (if OverviewWin
                       else                                      (* Make a new overview window.)
                            (SETQ OverviewWin (CREATEW (CREATEREGION (fetch (POSITION XCOORD) of
                                                                            NC.OffScreenPosition
                                                                     )
                                                                     (fetch (POSITION YCOORD) of NC.OffScreenPosition)
                                                                     OverviewWinWidth OverviewWinHeight)))
                            (WINDOWPROP BrowserWin 'BrowserOverviewWin OverviewWin))
                     (NC.ReattachBrowserOverviewWin OverviewWin BrowserWin WhereToAttach)
                     (NC.RedrawBrowserOverviewWin OverviewWin BrowserWin])

(NC.AskBrowserOverviewSpecs
  [LAMBDA (BrowserWin)                                           (* Randy.Gobbel " 4-Mar-87 14:00")

          (* * Put up stylesheet to get mode and where to attach overview win.)

          (* * rht 11/1/86%: Added NC.ProtectedCardOperation wrapper and check for ops in progress.)

          (* * rg |3/4/87| rewritten for new version of NC.ProtectedCardOperation, removed DontCheckOpInProgressFlg)

    (LET ((Card (NC.CoerceToCard BrowserWin)))
         (NC.ProtectedCardOperation Card "Browser Overview Specs" NIL
                (LET ((WhereLastAttached (OR (WINDOWPROP BrowserWin 'WHERETOATTACHOVERVIEWWIN)
                                             NC.DefaultWhereToAttachOverviewWin))
                      (LastMode (OR (WINDOWPROP BrowserWin 'OVERVIEWWINMODE)
                                    NC.DefaultBrowserOverviewMode))
                      OverviewSpecsResults)
                     (STYLE.PROP NC.BrowserOverviewSpecsStylesheet 'SELECTIONS (LIST (CAR WhereLastAttached)
                                                                                     (SELECTQ (CDR
                                                                                                WhereLastAttached
                                                                                                )
                                                                                         ((TOP RIGHT)
                                                                                              'TOP/RIGHT)
                                                                                         ((LEFT BOTTOM)
                                                                                              'BOTTOM/LEFT)
                                                                                         'CENTER)
                                                                                     LastMode))
                     [STYLE.PROP NC.BrowserOverviewSpecsStylesheet 'POSITION (create POSITION
                                                                                    XCOORD _
                                                                                    (fetch (REGION LEFT)
                                                                                       of (WINDOWPROP
                                                                                            BrowserWin
                                                                                            'REGION))
                                                                                    YCOORD _ (fetch (REGION TOP)
                                                                                                of (WINDOWREGION
                                                                                                    BrowserWin]
                     (if (SETQ OverviewSpecsResults (STYLESHEET NC.BrowserOverviewSpecsStylesheet))
                         then (WINDOWPROP BrowserWin 'WHERETOATTACHOVERVIEWWIN (CONS (CAR OverviewSpecsResults)
                                                                                     (CADR OverviewSpecsResults)
                                                                                     ))
                              (WINDOWPROP BrowserWin 'OVERVIEWWINMODE (CADDR OverviewSpecsResults))
                       else NIL])

)

(DEFINEQ

(NC.DRAWBOX
```

```
[LAMBDA (Left Bottom Width Height LineWidth Operation Stream)  (* rht%: "27-Feb-86 22:34")

          (* * This really SHOULD be in Interlisp. The one in Koto takes no Operation arg.)

    (DRAWLINE Left Bottom (PLUS Left Width)
           Bottom LineWidth Operation Stream)
    (DRAWLINE (PLUS Left Width)
           Bottom
           (PLUS Left Width)
           (PLUS Bottom Height)
           LineWidth Operation Stream)
    (DRAWLINE (PLUS Left Width)
           (PLUS Bottom Height)
           Left
           (PLUS Bottom Height)
           LineWidth Operation Stream)
    (DRAWLINE Left (PLUS Bottom Height)
           Left Bottom LineWidth Operation Stream])
```

(NC.ShrinkGraphToWindow

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

          (* * Bitblt's into Window a shrunken bitmap of the graph just big enough to fit.)

          (* * rht 3/18/86%: Now does more intelligent copy of Graph structure, preserving virtual node eq-ness.)

          (* * rht 3/21/86%: Now handles XScale and YScale.)

          (* * rg |11/19/86| Ref. to undef var BrowserWin --> GraphWindow
          (\, added) GLOBALVARS)

    (DECLARE (GLOBALVARS NC.OverviewWinMode.Expand))
    (LET ((GraphWindow (MAINWINDOW Window))
          (Scales (NC.ComputeOverviewScale Graph Window))
          (ShrunkenGraph (create GRAPH using Graph))
          Scale XScale YScale)
         [if (EQ (WINDOWPROP GraphWindow 'OVERVIEWWINMODE)
                 NC.OverviewWinMode.Expand)
             then (SETQ XScale (CAR Scales))
                  (SETQ YScale (CDR Scales))
                  (SETQ Scale (FMIN XScale YScale))
           else (SETQ Scale (SETQ XScale (SETQ YScale (FMIN (CAR Scales)
                                                            (CDR Scales]
                                                     (* Shrink the graphnodes.)
         (replace (GRAPH GRAPHNODES) of ShrunkenGraph with (for GraphNode in (fetch (GRAPH GRAPHNODES) of
                                                                                    ShrunkenGraph
                                                                                    )
                                                            eachtime (BLOCK)
                                                            collect (NC.ScaleGraphNode (create GRAPHNODE
                                                                                               using GraphNode)
                                                                                       Scale XScale YScale GraphWindow)))
                                                     (* Display the graph, but temporarily scale the arrowhead global
         vars. Will this work compiled?)
         (SHOWGRAPH ShrunkenGraph Window)
         (WINDOWPROP Window 'Scale Scale)
         (WINDOWPROP Window 'XScale XScale)
         (WINDOWPROP Window 'YScale YScale)
         (WINDOWPROP Window 'GRAPH ShrunkenGraph])
```

(NC.ScaleGraphNode

```
[LAMBDA (GraphNode Scale XScale YScale BrowserWin)              ; Edited 29-Apr-88 11:52 by pmi
    ;; Change the position and node label according to Scale.
    ;; rht 3/14/86: Now smashes NODELABEL field of graph node with bitmap rather than NODELABELBITMAP field.
    ;; rht 3/18/86: Now returns GraphNode.
    ;; rht 3/21/86: Now scales position using XScale and YScale.  But note that new node size still governed by Scale.  This is because SCALEBITMAP
    ;; only takes single scale.
    ;; pmi 4/29/88: Now uses the DisplayFn of the ImageObj instead of calling NC.LinkIconDisplayFn, which didn't work for Deleted Link ImageObj's.
    (LET ((OldWidth (fetch (GRAPHNODE NODEWIDTH) of GraphNode))
          (OldHeight (fetch (GRAPHNODE NODEHEIGHT) of GraphNode))
          (OldPosition (fetch (GRAPHNODE NODEPOSITION) of GraphNode))
          NewWidth NewHeight NewBitmap OldBitmap NodeLabel)
         [replace (GRAPHNODE NODEWIDTH) of GraphNode with (SETQ NewWidth (FIX (FTIMES Scale OldWidth]
         [replace (GRAPHNODE NODEHEIGHT) of GraphNode with (SETQ NewHeight (FIX (FTIMES Scale OldHeight]
         [replace (GRAPHNODE NODEPOSITION) of GraphNode with (create POSITION
                                                                     XCOORD _ (FIX (FTIMES XScale
                                                                                           (fetch (POSITION XCOORD)
                                                                                              of OldPosition)))
                                                                     YCOORD _ (FIX (FTIMES YScale
                                                                                           (fetch (POSITION YCOORD)
                                                                                              of OldPosition]
         (if (LESSP Scale NC.LeastScaleForGraphNodeShrinking)
             then                                               ; At this scale no need to shrink node label.  Just show shaded
                                                                ; box.
```

```
                (SETQ NewBitmap (BITMAPCREATE NewWidth NewHeight))
                (BLTSHADE GRAYSHADE NewBitmap NIL NIL NewWidth NewHeight)
          else                                                    ; Shrink node label.
               [if (SETQ OldBitmap (fetch (GRAPHNODE NODELABELBITMAP) of GraphNode))
                 else (SETQ OldBitmap (BITMAPCREATE OldWidth OldHeight))
                      (SETQ NodeLabel (fetch (GRAPHNODE NODELABEL) of GraphNode))
                                                              ; Apply the appropriate DisplayFn to the imageobj
                      (if (IMAGEOBJP NodeLabel)
                          then (APPLY* (IMAGEOBJPROP NodeLabel 'DISPLAYFN)
                                       NodeLabel
                                       (DSPCREATE OldBitmap))
                        else (PRIN1 NodeLabel (DSPCREATE OldBitmap]
               (SETQ NewBitmap (SCALEBITMAP OldBitmap Scale)))
         (replace (GRAPHNODE NODELABEL) of GraphNode with NewBitmap)
         (replace (GRAPHNODE NODELABELBITMAP) of GraphNode with NIL)
         GraphNode])
```

(NC.ComputeOverviewScale

```
[LAMBDA (Graph OverviewWin)                                   (* rht%: "21-Mar-86 00:13")
```

(* * Figure out the scale necessary to shrink Graph into OverviewWin.)

(* * rht 3/21/86%: Now returns both XScale and YScale.)

```
   (LET ((GraphRegion (GRAPHREGION Graph)))
        (CONS (FMIN (FQUOTIENT (WINDOWPROP OverviewWin 'WIDTH)
                               (fetch (REGION WIDTH) of GraphRegion))
                    1)
              (FMIN (FQUOTIENT (WINDOWPROP OverviewWin 'HEIGHT)
                               (fetch (REGION HEIGHT) of GraphRegion))
                    1])
```

(NC.RedrawBrowserOverviewWin

```
[LAMBDA (OverviewWin BrowserWin)                              (* rht%: " 8-May-86 11:29")
```

(* * The contents of the browser window have changed. Redraw the overview win.)

(* * rht 3/20/86%: Now compresses to fit overview if necessary.)

(* * rht 5/8/86%: Now smashes SCROLLFN)

```
   (NC.ShrinkGraphToWindow (WINDOWPROP BrowserWin 'GRAPH)
          OverviewWin)
   (if (EQ (WINDOWPROP BrowserWin 'OVERVIEWWINMODE)
          NC.OverviewWinMode.Compress)
       then (WINDOWPROP OverviewWin 'RESHAPEFN NIL)
            (NC.CompressOverviewWin OverviewWin BrowserWin))
   (WINDOWPROP OverviewWin 'REPAINTFN (FUNCTION NC.BrowserOverviewWinRepaintFn))
   (WINDOWPROP OverviewWin 'RESHAPEFN (FUNCTION NC.BrowserOverviewWinReshapeFn))
   (WINDOWPROP OverviewWin 'BUTTONEVENTFN (FUNCTION NC.BrowserOverviewWinButtonEventFn))
   (WINDOWPROP OverviewWin 'SCROLLFN NIL)
   (NC.DrawWireFrameInOverviewWin OverviewWin BrowserWin])
```

(NC.DrawWireFrameInOverviewWin

```
[LAMBDA (OverviewWin BrowserWin)                              (* rht%: "21-Mar-86 00:35")
```

(* * Draw a wire frame in the overview win whose size and position within the overview correspond to the size and position of BrowserWin relative to entire graph.)

(* * rht 3/21/86%: Now handles XScale and YScale.)

```
   (LET ((BrowserClippingRegion (DSPCLIPPINGREGION NIL BrowserWin))
         (XScale (WINDOWPROP OverviewWin 'XScale))
         (YScale (WINDOWPROP OverviewWin 'YScale))
         WireFrame)
        [SETQ WireFrame (CREATEREGION (FIX (FTIMES XScale (fetch (REGION LEFT) of BrowserClippingRegion)))
                                      (FIX (FTIMES YScale (fetch (REGION BOTTOM) of BrowserClippingRegion)))
                                      (FIX (FTIMES XScale (fetch (REGION WIDTH) of BrowserClippingRegion)))
                                      (FIX (FTIMES YScale (fetch (REGION HEIGHT) of BrowserClippingRegion]
                                                              (* Draw the wire frame out on the overview window.)
        (NC.DRAWBOX (fetch (REGION LEFT) of WireFrame)
               (fetch (REGION BOTTOM) of WireFrame)
               (fetch (REGION WIDTH) of WireFrame)
               (fetch (REGION HEIGHT) of WireFrame)
               1
               'INVERT OverviewWin)                           (* Stash the new wire frame on windowprop.)
        (WINDOWPROP OverviewWin 'LastWireFrame WireFrame])
```

(NC.CompressOverviewWin

```
[LAMBDA (OverviewWin GraphWin)                                (* rht%: "21-Mar-86 16:03")
```

(* * Reshape OverviewWin so it just holds its contents plus the wire frame.)

```
          (* * rht 3/21/86%: Massive rewrite. Notice ugly use of constant 7 in width of NewRegion.
          This seems necessary empirically to keep overview from shrinking with successive redisplays.)

    (LET ((CurCLIPPINGRegion (DSPCLIPPINGREGION NIL OverviewWin))
          (CurRegion (WINDOWREGION OverviewWin))
          (WireFrameRegion (WINDOWPROP OverviewWin 'LastWireFrame))
          [GraphRegion (GRAPHREGION (WINDOWPROP OverviewWin 'GRAPH]
          (OverviewWinBorder (WINDOWPROP OverviewWin 'BORDER))
          NewRegion)
         [SETQ NewRegion (create REGION using GraphRegion WIDTH _ (PLUS 7 (fetch (REGION WIDTH) of GraphRegion]
         (if [OR (NULL WireFrameRegion)
                 (AND (NOT (EQUAL NewRegion CurCLIPPINGRegion))
                      (REGIONSINTERSECTP WireFrameRegion NewRegion))
                 (PROG1 (SUBREGIONP CurCLIPPINGRegion WireFrameRegion)
                     (SETQ NewRegion (EXTENDREGION NewRegion (INTERSECTREGIONS WireFrameRegion CurCLIPPINGRegion)
                                                   )))]
             then [SHAPEW OverviewWin (CREATEREGION (fetch (REGION LEFT) of CurRegion)
                                                    (fetch (REGION BOTTOM) of CurRegion)
                                                    (PLUS (TIMES 2 OverviewWinBorder)
                                                          (fetch (REGION WIDTH) of NewRegion))
                                                    (PLUS (TIMES 2 OverviewWinBorder)
                                                          (fetch (REGION HEIGHT) of NewRegion]
                  (NC.ReattachBrowserOverviewWin OverviewWin GraphWin (WINDOWPROP GraphWin
                                                                                  'WHERETOATTACHOVERVIEWWIN])
```

(NC.ReattachBrowserOverviewWin

```
  [LAMBDA (OverviewWin BrowserWin WhereToAttach)                    (* rht%: "24-Jul-87 19:10")

          (* * Check to see if OverviewWin is attached at correct place. If not, detach and reattach.)

          (* * rht 3/7/86%: No longer checks. Just reattaches.)

          (* * rht 4/2/87%: Changed WINDOWDELPROP of PASSTOMAINCOMS to just smash them.
          Otherwise Shrager's special SHAPEW.POP, etc. crud hangs around.)

          (* * rht 7/24/87%: Took SHAPEW off of list of PASSTOMAINCOMS.)

    (LET [(Width (fetch (REGION WIDTH) of (WINDOWREGION OverviewWin)))
          (Height (fetch (REGION HEIGHT) of (WINDOWREGION OverviewWin]
         (DETACHWINDOW OverviewWin)
         (ATTACHWINDOW OverviewWin BrowserWin (OR (CAR WhereToAttach)
                                                  'LEFT)
                (OR (CDR WhereToAttach)
                    'TOP)
                'LOCALCLOSE)
         (WINDOWPROP OverviewWin 'PASSTOMAINCOMS '(MOVEW SHRINKW BURYW))
         (WINDOWPROP BrowserWin 'OverviewWinWidth Width)
         (WINDOWPROP BrowserWin 'OverviewWinHeight Height)
         (WINDOWPROP OverviewWin 'MINSIZE (FUNCTION NC.BrowserOverviewWinMINSIZEFn))
         (WINDOWPROP OverviewWin 'MAXSIZE (CONS Width Height])
```

(NC.BrowserScrollFn

```
  [LAMBDA (WINDOW XDELTA YDELTA CONTINUOUSFLG)                      (* rht%: "27-Feb-86 22:49")

          (* * If there's an overview win, then update its wire frame.)

    (LET [(OverviewWin (WINDOWPROP WINDOW 'BrowserOverviewWin]
         (if (OPENWP OverviewWin)
             then (NC.DrawWireFrameInOverviewWin OverviewWin WINDOW)
                  (SCROLLBYREPAINTFN WINDOW XDELTA YDELTA CONTINUOUSFLG)
                  (NC.DrawWireFrameInOverviewWin OverviewWin WINDOW)
           else (SCROLLBYREPAINTFN WINDOW XDELTA YDELTA CONTINUOUSFLG])
```

(NC.BrowserReshapeFn

```
  [LAMBDA (Window OldWinBitmap OldWinRegion)                        (* rht%: "27-Feb-86 22:48")

          (* * If there's an overview win, then update its wire frame.)

    (LET [(OverviewWin (WINDOWPROP Window 'BrowserOverviewWin]
         (if (OPENWP OverviewWin)
             then (NC.DrawWireFrameInOverviewWin OverviewWin Window)
                  (RESHAPEBYREPAINTFN Window OldWinBitmap OldWinRegion)
                  (NC.DrawWireFrameInOverviewWin OverviewWin Window)
           else (RESHAPEBYREPAINTFN Window OldWinBitmap OldWinRegion])
```

(NC.BrowserOverviewWinRepaintFn

```
  [LAMBDA (OverviewWin Region)                                      (* rht%: " 3-Mar-86 17:57")

          (* * Recomputes overview.)

    (NC.RedrawBrowserOverviewWin OverviewWin (MAINWINDOW OverviewWin])
```

(NC.BrowserOverviewWinReshapeFn

```
[LAMBDA (OverviewWin OldWinBitmap OldWinRegion)                    (* rht%: "20-Mar-87 14:33")

        (* * When overview win is reshaped, we need to recompute its contents from scratch.
        We detach and reattach in case user pulled it away from browser edge.)

        (* * rht 3/20/87%: Now instead of deleting SHAPEW from PASSTOMAINCOMS, it replaces whole list.
        This is because Shrager's WDWHACKS package adds screwy SHAPEW.mumble things to PASSTOMAINCOMS.)

    (LET [(BrowserWin (MAINWINDOW OverviewWin))
          (WhereAttached (WINDOWPROP OverviewWin 'WHEREATTACHED]
         (DETACHWINDOW OverviewWin)
         (ATTACHWINDOW OverviewWin BrowserWin (OR (CAR WhereAttached)
                                                  'LEFT)
                (OR (CDR WhereAttached)
                    'TOP)
                'LOCALCLOSE)
         (WINDOWPROP OverviewWin 'PASSTOMAINCOMS '(MOVEW SHRINKW BURYW))
         (WINDOWPROP BrowserWin 'OverviewWinWidth (fetch (REGION WIDTH) of (WINDOWREGION OverviewWin)))
         (WINDOWPROP BrowserWin 'OverviewWinHeight (fetch (REGION HEIGHT) of (WINDOWREGION OverviewWin)))
         [WINDOWPROP OverviewWin 'MAXSIZE (CONS (fetch (REGION WIDTH) of (WINDOWREGION OverviewWin))
                                                (fetch (REGION HEIGHT) of (WINDOWREGION OverviewWin]
         (NC.RedrawBrowserOverviewWin OverviewWin BrowserWin])
```

(NC.BrowserOverviewWinMINSIZEFn

```
[LAMBDA (OverviewWin)                                              (* rht%: "21-Mar-86 15:17")

        (* * This horrible kludge is so that ATTACHEDWINDOWS sees a different value for MINSIZE depending on whether we're
        reshaping the whole browser or just the overview win. In the former case, we want the MINSIZE to be the current size of the
        overview win. In the latter, we want it at some small value so user can reduce size of overview win.)

    (LET (Stkpos)
         (PROG1 (if (OR [NULL (SETQ Stkpos (STKPOS 'SHAPEW]
                        (NEQ OverviewWin (STKARG 'WINDOW 'SHAPEW))
                        (SETQ Stkpos (STKPOS 'RESHAPEALLWINDOWS NIL NIL Stkpos)))
                    then [LET ((BrowserWin (MAINWINDOW OverviewWin)))
                              (CONS (WINDOWPROP BrowserWin 'OverviewWinWidth)
                                    (WINDOWPROP BrowserWin 'OverviewWinHeight]
                  else '(5 . 5))
                (RELSTK Stkpos])
```

(NC.BrowserOverviewWinButtonEventFn

```
[LAMBDA (OverviewWin)                                              (* rht%: "21-Mar-86 00:43")

        (* * Called when user buttons in browser overview window. bring cursor to nearest corner of wire frame and let him
        reposition wire frame.)

        (* * rht |3/15/86| Now allows moves of wire frame that increase scrolling region.)

        (* * rht 3/21/86%: Now handles XScale and YScale.)

    (LET ((LastWireFrame (WINDOWPROP OverviewWin 'LastWireFrame))
          NewRegion XScaleReciprocal YScaleReciprocal BrowserWin OldDisplayRegion NewLeft NewBottom)
         (SETQ NewRegion (GETBOXREGION (fetch (REGION WIDTH) of LastWireFrame)
                                       (fetch (REGION HEIGHT) of LastWireFrame)
                                       (PLUS (DSPXOFFSET NIL OverviewWin)
                                             (fetch (REGION LEFT) of LastWireFrame))
                                       (PLUS (DSPYOFFSET NIL OverviewWin)
                                             (fetch (REGION BOTTOM) of LastWireFrame))
                                      OverviewWin "Please position browser overview wire frame."))
         (if (REGIONSINTERSECTP NewRegion (DSPCLIPPINGREGION NIL OverviewWin))
             then (SETQ BrowserWin (MAINWINDOW OverviewWin))
                  [SETQ XScaleReciprocal (FQUOTIENT 1 (WINDOWPROP OverviewWin 'XScale]
                  [SETQ YScaleReciprocal (FQUOTIENT 1 (WINDOWPROP OverviewWin 'YScale]
                                                                   (* Where browser used to be.)
                  (SETQ OldDisplayRegion (DSPCLIPPINGREGION NIL BrowserWin))
                                                                   (* First make sure browser win's EXTENT is big enough.)
                  [WINDOWPROP BrowserWin 'EXTENT (UNIONREGIONS (WINDOWPROP BrowserWin 'EXTENT)
                                                               (CREATEREGION (SETQ NewLeft
                                                                                   (FIX (FTIMES (fetch (REGION LEFT)
                                                                                                   of NewRegion)
                                                                                                XScaleReciprocal)))
                                                                             (SETQ NewBottom (FIX (FTIMES (fetch (REGION BOTTOM
                                                                                                                           )
                                                                                                            of NewRegion)
                                                                                                         YScaleReciprocal)))
                                                                             (fetch (REGION WIDTH) of OldDisplayRegion)
                                                                             (fetch (REGION HEIGHT) of OldDisplayRegion]
                                                                   (* Scroll browser to new region.)
                  (SCROLLW BrowserWin (DIFFERENCE (fetch (REGION LEFT) of OldDisplayRegion)
                                                  NewLeft)
                           (DIFFERENCE (fetch (REGION BOTTOM) of OldDisplayRegion)
                                       NewBottom])
```

(NC.BrowserCardQuitFn

```
[LAMBDA (Card)                                          ; Edited 15-Jan-88 20:00 by Trigg
    ;; This clears all UserData fields of Graph node UIDs.  I ONLY HAVE TO DO THIS BECAUSE INTERLISP WON'T GC CYCLES!
    ;; rht 1/15/88: Had to replace the NCP.ApplySuperTypeFn call with grungy apply* to prevent infinite recursive calls.
    [for GraphNode in (fetch (GRAPH GRAPHNODES) of (NC.FetchSubstance Card))
       do (LET ((GraphNodeID (NC.CoerceToGraphNodeIDOrLabel GraphNode)))
               (AND (type? UID GraphNodeID)
                    (NC.UIDSetPropList GraphNodeID NIL]
    (APPLY* (NCP.CardTypeFn 'Graph 'QuitFn)
           Card])
```

(NC.MakeBrowserCardReadOnly

```
[LAMBDA (Card)                                          ; Edited 27-Jan-88 14:54 by MacDonald

        (* * Make a BrowserCard Read-Only.)

    (DECLARE (GLOBALVARS NC.ShowPropListMenu))
    (LET ((Window (NC.FetchWindow Card))
          PropListEditor)
         (NC.ProtectedCardOperation Card "Make Read-Only" Window (if (NC.CardSomehowDirtyP Card)
                                                                     then (NC.SaveOrRevertGraphCard Card Window)
                                                                    )
                (NC.SetUserDataProp Card 'ReadOnly T)
                (NC.GetGraphEditMenu Window)
                (SETQ NC.BrowserSafeItems '(Show% Links Show% Info Designate% FileBoxes |Close and Save|
                                                |Browser Overview Win|))
                (SETQ NC.BrowserSafeSubItems '(|Close and Save| |Close w/o Saving| |Save in NoteFile|
                                                Indicate% NoteFile))
                (if (WINDOWP (SETQ PropListEditor (NC.PropListEditorOpenP Window)))
                    then (NC.MakeTEditReadOnly Window)
                         (WINDOWPROP PropListEditor 'TEDIT.MENU NC.ShowPropListMenu))
                (NC.MakeMenusReadOnly Window NC.BrowserSafeItems NC.BrowserSafeSubItems])
```

(NC.MakeBrowserCardReadWrite

```
[LAMBDA (Card)                                          ; Edited 27-Jan-88 14:56 by MacDonald

        (* * dwm |1/6/88| coerce Read/Write on a browser card)

    (DECLARE (GLOBALVARS NC.EditPropListMenu))
    (LET ((Window (NC.FetchWindow Card))
          PropListEditor)
         (NC.ProtectedCardOperation Card "Make Read-Write" Window (if (WINDOWP (SETQ PropListEditor (
                                                                                NC.PropListEditorOpenP
                                                                                  Window)))
                                                                  then (NC.MakeTEditReadWrite PropListEditor
                                                                         )
                                                                       (WINDOWPROP PropListEditor
                                                                              'TEDIT.MENU
                                                                              NC.EditPropListMenu))
                (NC.SetUserDataProp Card 'ReadOnly NIL)
                (WINDOWPROP Window 'RIGHTBUTTONFN (FUNCTION NC.BrowserRightButtonFn))
                (NC.MakeMenusReadWrite Window])

)


;;; Miscellaneous

(DEFINEQ

(NC.DelReferencesToCardFromBrowser
  [LAMBDA (SourceCard LinkOrDestinationCard Don'tCreateDeletedImageObjFlg)
                                                        ; Edited 28-Mar-89 13:24 by SYE.ENVOS

        (* * Delete from the browser specified by SourceCard all link icon nodes whose DESTINATIONID is eq to DestinationID.
        This just checks the case of the SourceCard being a browser root and then passes off to GRAPHCARD's DelReferencesFn.)

        (* * rht 4/30/86%: No longer passes control up to Super's DeleteLinksFn.
        Work is now done here.)

        (* * rht 9/2/86%: Now sets dirtyflg of substance if change was made.)

        (* * rht 11/4/86%: Now takes Don'tCreateDeletedImageObjFlg arg.)

        (* * rht 6/10/87%: Now cleans up better in case when we're replacing node by deleted link icon.)
    ;; kms 3/28/89: set NC.BrowserRemoveNode's argument Don'tConfirmFlg to be true
    ;;
    (DECLARE (GLOBALVARS NC.UseDeletedLinkIconIndicatorsFlg NC.DeletedLinkImageObject))
    (LET ((LinkFlg (type? Link LinkOrDestinationCard))
          (ImageBox (NC.DeletedLinkImageBoxFn NC.DeletedLinkImageObject))
          LinkIcon Graph DestinationCard BrowserRoots RootCardToDelete UID)
         (if LinkFlg
```

```
            then (OR (NC.CardP SourceCard)
                     (SETQ SourceCard (fetch (Link SourceCard) of LinkOrDestinationCard)))
                 (SETQ DestinationCard (fetch (Link DestinationCard) of LinkOrDestinationCard))
          else (SETQ DestinationCard LinkOrDestinationCard))
        (if [SETQ RootCardToDelete (for RootCard in (SETQ BrowserRoots (NC.FetchBrowserRoots SourceCard))
                                      eachtime (BLOCK) do (if (NC.SameCardP DestinationCard RootCard)
                                                              then (RETURN RootCard]
            then (NC.SetBrowserRoots SourceCard (DREMOVE RootCardToDelete BrowserRoots)))
        (SETQ Graph (NC.FetchSubstance SourceCard))
        (for GraphNode in (fetch (GRAPH GRAPHNODES) of Graph)
           when (AND (NC.LinkIconImageObjP (SETQ LinkIcon (fetch (GRAPHNODE NODELABEL) of GraphNode)))
                     (if LinkFlg
                         then (NC.SameLinkP LinkOrDestinationCard (NC.FetchLinkFromLinkIcon LinkIcon))
                       else (NC.SameCardP (fetch (Link DestinationCard) of (NC.FetchLinkFromLinkIcon LinkIcon))
                                          DestinationCard)))
           do (if (AND NC.UseDeletedLinkIconIndicatorsFlg (NOT Don'tCreateDeletedImageObjFlg))
                  then (replace (GRAPHNODE NODELABEL) of GraphNode with NC.DeletedLinkImageObject)
                       (replace (GRAPHNODE NODEWIDTH) of GraphNode with (fetch (IMAGEBOX XSIZE) of ImageBox))
                       (replace (GRAPHNODE NODEHEIGHT) of GraphNode with (fetch (IMAGEBOX YSIZE) of ImageBox))
                       (NC.RemoveBrowserNodeHashArrayEntry SourceCard DestinationCard)
                       (if (type? UID (SETQ UID (fetch (GRAPHNODE NODEID) of GraphNode)))
                           then (NC.UIDSetPropList UID NIL))
                       (NC.SetSubstanceDirtyFlg SourceCard T)
                else (NC.BrowserRemoveNode Graph (NC.FetchWindow SourceCard)
                                           NIL GraphNode T T)))
        (if (AND (NC.ActiveCardP SourceCard)
                 (NC.FetchWindow SourceCard))
            then (REDISPLAYGRAPH (NC.FetchWindow SourceCard])
```

(NC.NewBrowserNodeUIDFromOldUID

```
[LAMBDA (OldUID GraphNodes HashArray)                    (* rht%: " 1-Feb-86 15:59")
```

(* * Find and return the NodeID UID in GraphNodes that is SameUIDP to OldUID.
Use hash table for caching.)

```
  (OR (GETHASH OldUID HashArray)
      (for GraphNode in GraphNodes bind NewUID eachtime (BLOCK) when (NC.SameUIDP OldUID (SETQ NewUID
                                                                                          (fetch (GRAPHNODE
                                                                                                   NODEID)
                                                                                            of GraphNode)))
       do (RETURN (PUTHASH OldUID NewUID HashArray])
```

(NC.GetBrowserSubstance

```
[LAMBDA (Card Length Stream VersionNum)                  (* rht%: " 4-May-87 20:16")
```

(* * Go get all the browser-specific info and then get the graph that is the browser's substance.)

(* * rht 2/14/86%: Added call to NC.ApplySupersFn)

(* * rht 2/28/86%: Added special handling for old version -1 style.
I.e. pre 1.3k.)

(* * fgh |5/1/86| Can't have negative versions, so changed old style to be 255 255.0 Also added defaults for old style browser specs.)

(* * fgh |5/25/86| Undid preceeding change w.r.t. the 255 versus -1 Turns out -1 never came from file, only from converter as an arg.)

(* * rht 5/4/87%: Changed call from NC.ApplySupersFn to APPLY* because otherwise we generate stack overflow for cards that specialize browser card type.)

```
  (if (NOT (EQP VersionNum -1))
      then (NC.SetBrowserRootsInfo Card (NC.ReadBrowserRootsInfo Stream))
           (NC.SetBrowserLinkLabels Card (NC.ReadBrowserLinkLabels Stream))
           (NC.SetBrowserFormat Card (NC.ReadBrowserFormat Stream))
           (NC.SetSpecialBrowserSpecs Card (NC.ReadSpecialBrowserSpecs Stream))
           (NC.SetBrowserDepth Card (NC.ReadBrowserDepth Stream))
           (NC.SetBrowserLinksLegend Card (NC.ReadBrowserLinksLegend Stream))
           (NC.SetBrowserSavedLinkingInfo Card (NC.ReadBrowserSavedLinkingInfo Stream))
    else (NC.SetBrowserRootsInfo Card NIL)
         (NC.SetBrowserLinkLabels Card NIL)
         (NC.SetBrowserFormat Card NIL)
         (NC.SetSpecialBrowserSpecs Card NIL)
         (NC.SetBrowserDepth Card 0)
         (NC.SetBrowserLinksLegend Card NIL)
         (NC.SetBrowserSavedLinkingInfo Card NIL))
  (APPLY* (NCP.CardTypeFn 'Graph 'GetFn)
          Card Length Stream VersionNum])
```

(NC.ComputeBrowserSavedLinkingInfo

```
[LAMBDA (Card)                                           (* rht%: "30-Apr-86 15:30")
```

(* * Search the graph nodeIDs of Card's substance for special linking info for multiple links between same pair of nodes.)

```
    (for GraphNode in (fetch (GRAPH GRAPHNODES) of (NC.FetchSubstance Card)) bind SavedLinkingInfoForNode UID
       eachtime (BLOCK) when (AND (type? UID (SETQ UID (fetch (GRAPHNODE NODEID) of GraphNode)))
                                  (SETQ SavedLinkingInfoForNode (NC.ComputeBrowserSavedLinkingInfoForNode UID)))
       collect (CONS UID SavedLinkingInfoForNode])
```

(NC.ComputeBrowserSavedLinkingInfoForNode

```
  [LAMBDA (UID)                                                    (* rht%: " 7-Feb-86 00:43")

          (* * Return list of DestUIDs and link dashing info.)

    (for UserData on (NC.UIDGetPropList UID) by (CDDR UserData) when (type? UID (CAR UserData))
       join (LIST (CAR UserData)
                  (CADR UserData])

)

(DEFINEQ
```

(NC.FetchBrowserRootsInfo

```
  [LAMBDA (Card)                                                   (* rht%: "11-Feb-86 23:28")

          (* * Return the browser roots UID pairs for this browser card.)

    (NC.FetchUserDataProp Card 'BrowserRootsInfo])
```

(NC.FetchBrowserLinkLabels

```
  [LAMBDA (Card)                                                   (* rht%: "11-Feb-86 23:28")

          (* * Return the browser link labels for this browser card.)

    (NC.FetchUserDataProp Card 'BrowserLinkLabels])
```

(NC.FetchBrowserFormat

```
  [LAMBDA (Card)                                                   (* rht%: "11-Feb-86 23:28")

          (* * Return the browser format for this browser card.)

    (NC.FetchUserDataProp Card 'BrowserFormat])
```

(NC.FetchSpecialBrowserSpecs

```
  [LAMBDA (Card)                                                   (* rht%: "11-Feb-86 23:28")

          (* * Return the special browser specs for this browser card.)

    (NC.FetchUserDataProp Card 'SpecialBrowserSpecs])
```

(NC.FetchBrowserDepth

```
  [LAMBDA (Card)                                                   (* rht%: "11-Feb-86 23:27")

          (* * Return the browser depth for this browser card.)

    (NC.FetchUserDataProp Card 'BrowserDepth])
```

(NC.FetchBrowserSavedLinkingInfo

```
  [LAMBDA (Card)                                                   (* rht%: "11-Feb-86 23:27")

          (* * Return the special linking info for this browser card.)

    (NC.FetchUserDataProp Card 'BrowserSavedLinkingInfo])
```

(NC.FetchBrowserLinksLegend

```
  [LAMBDA (Card)                                                   (* rht%: "11-Feb-86 23:27")

          (* * Return the links legend label pairs for this browser card.)

    (NC.FetchUserDataProp Card 'BrowserLinksLegend])

)

(DEFINEQ
```

(NC.SetBrowserRootsInfo

```
  [LAMBDA (Card BrowserRootsInfo)                                  (* rht%: "11-Feb-86 23:30")

          (* * Set the browser card's browser roots pairs.)

    (NC.SetUserDataProp Card 'BrowserRootsInfo BrowserRootsInfo])
```

(NC.SetBrowserLinkLabels

```
  [LAMBDA (Card BrowserLinkLabels)                                    (* rht%: "11-Feb-86 23:30")

          (* * Set the browser card's link labels.)

    (NC.SetUserDataProp Card 'BrowserLinkLabels BrowserLinkLabels])
```

(NC.SetBrowserFormat

```
  [LAMBDA (Card BrowserFormat)                                        (* rht%: "11-Feb-86 23:30")

          (* * Set the browser card's format.)

    (NC.SetUserDataProp Card 'BrowserFormat BrowserFormat])
```

(NC.SetSpecialBrowserSpecs

```
  [LAMBDA (Card SpecialBrowserSpecs)                                  (* rht%: "11-Feb-86 23:30")

          (* * Set the browser card's special browser specs.)

    (NC.SetUserDataProp Card 'SpecialBrowserSpecs SpecialBrowserSpecs])
```

(NC.SetBrowserDepth

```
  [LAMBDA (Card BrowserDepth)                                         (* rht%: "11-Feb-86 23:29")

          (* * Set the browser card's depth.)

    (NC.SetUserDataProp Card 'BrowserDepth BrowserDepth])
```

(NC.SetBrowserSavedLinkingInfo

```
  [LAMBDA (Card BrowserSavedLinkingInfo)                              (* rht%: "20-Feb-86 11:44")

          (* * Set the browser special linking info.)

          (* * rht 2/20/86%: Now returns the value we set to.)

    (NC.SetUserDataProp Card 'BrowserSavedLinkingInfo BrowserSavedLinkingInfo)
   BrowserSavedLinkingInfo])
```

(NC.SetBrowserLinksLegend

```
  [LAMBDA (Card BrowserLinksLegend)                                   (* rht%: "11-Feb-86 23:29")

          (* * Set the browser card's links legend label pairs.)

    (NC.SetUserDataProp Card 'BrowserLinksLegend BrowserLinksLegend])

)

(DEFINEQ
```

(NC.ReadBrowserRootsInfo

```
  [LAMBDA (Stream)                                                    (* rht%: "20-Feb-86 14:34")

          (* * Read pairs of UIDs corresponding to card and notefile UIDs for each browser root.
          These pairs are bounded by left and right parens.)

    (LET (CharRead)
         (if (NEQ (SETQ CharRead (CHARACTER (BIN Stream)))
                  '%(
             then (NC.ReportError "NC.ReadBrowserRootsInfo" (CONCAT "Expected to read left paranthesis. Saw: "
                                                                    CharRead)))
         (PROG1 (while (NEQ (CHARACTER (\PEEKBIN Stream))
                            '%))
                   eachtime (BLOCK) collect (CONS (NC.ReadUID Stream)
                                                  (NC.ReadUID Stream)))
                (BIN Stream])
```

(NC.ReadBrowserLinkLabels

```
  [LAMBDA (Stream)                                                    (* rht%: " 1-Nov-86 15:43")

          (* * Read list of link labels from notefile.)

          (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PROG1 (READ Stream NC.OrigReadTable)
           (BIN Stream])
```

(NC.ReadBrowserFormat

```
  [LAMBDA (Stream)                                                    (* rht%: " 1-Nov-86 15:43")

          (* * Read browser format from notefile.)
```

```
        (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PROG1 (READ Stream NC.OrigReadTable)
           (BIN Stream])
```

(NC.ReadSpecialBrowserSpecs

```
  [LAMBDA (Stream)                                                    (* rht%: " 1-Nov-86 15:44")

        (* * Read special browser specs from notefile.)

        (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PROG1 (READ Stream NC.OrigReadTable)
           (BIN Stream])
```

(NC.ReadBrowserDepth

```
  [LAMBDA (Stream)                                                    (* rht%: " 1-Nov-86 15:44")

        (* * Read depth from notefile.)

        (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET ((ThingRead (READ Stream NC.OrigReadTable)))
         (if (NOT (FIXP ThingRead))
             then (NC.ReportError "NC.ReadBrowserFormat" (CONCAT "Expected to read a number.  Saw: " ThingRead))
           )                                                          (* Skip CR)
         (BIN Stream)
         ThingRead])
```

(NC.ReadBrowserSavedLinkingInfo

```
  [LAMBDA (Stream)                                                    (* rht%: "20-Feb-86 14:36")

        (* * Read lists of info on multiple links between nodes. Each list consists of a UID cons'ed to a prop list of destination UIDs
        and link dashing info.)

    (LET (CharRead)
         (if (NEQ (SETQ CharRead (CHARACTER (BIN Stream)))
                  '%()
             then (NC.ReportError "NC.ReadBrowserSavedLinkingInfo" (CONCAT "Expected to read left paranthesis.
                                                                           Saw: " CharRead)))
         (PROG1 (while (NEQ (CHARACTER (\PEEKBIN Stream))
                            '%))
                   eachtime (BLOCK) collect (NC.ReadBrowserSavedLinkingInfoForNode Stream))
                                                                      (* Skip RightParen)
             (BIN Stream))])
```

(NC.ReadBrowserSavedLinkingInfoForNode

```
  [LAMBDA (Stream)                                                    ; Edited 11-May-88 22:14 by Trigg

;;; Read list of info on multiple links for a particular node.  In form of a source UID cons'ed to a prop list of destination UIDs and link dashing info.

    ;; rht 11/1/86: Now uses our readtable when reading.

    ;; rht 5/11/88: Now passes non-nil second arg to NC.ReadUID so as to reuse old equivalent UID objects if any.

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (LET (CharRead SourceUID)
         (if (NEQ (SETQ CharRead (CHARACTER (BIN Stream)))
                  '%()
             then (NC.ReportError "NC.ReadBrowserSavedLinkingInfoForNode" (CONCAT "Expected to read left
                                                                                  paranthesis. Saw: " CharRead)))
         (PROG1 [CONS (NC.ReadUID Stream T)
                      (while (NEQ (CHARACTER (\PEEKBIN Stream))
                                  '%))
                         eachtime (BLOCK) join (PROG1 (LIST (NC.ReadUID Stream T)
                                                            (READ Stream NC.OrigReadTable))
                                                      ; Skip CR
                                               (BIN Stream))] ; Skip RightParen
             (BIN Stream))])
```

(NC.ReadBrowserLinksLegend

```
  [LAMBDA (Stream)                                                    (* rht%: " 1-Nov-86 15:44")

        (* * Read links legend from notefile.)

        (* * rht 11/1/86%: Now uses our readtable when reading.)

    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PROG1 (READ Stream NC.OrigReadTable)
           (BIN Stream])
```

)

(DEFINEQ

(**NC.WriteBrowserRootsInfo**
[LAMBDA (Stream BrowserRootUIDPairs) (* rht%: " 6-Feb-86 23:30")

(* * Write pairs of UIDs corresponding to card and notefile UIDs for each browser root.
These pairs are bounded by left and right parens.)

(PRIN1 '%(Stream)
(**for** BrowserRootUIDPair **in** BrowserRootUIDPairs **eachtime** (BLOCK) **do** (NC.WriteUID Stream (CAR BrowserRootUIDPair
))
(NC.WriteUID Stream (CDR BrowserRootUIDPair
)))
(PRIN1 '%) Stream])

(**NC.WriteBrowserLinkLabels**
[LAMBDA (Stream LinkLabels) (* rht%: " 1-Nov-86 15:37")

(* * Write list of link labels from notefile.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

(**DECLARE** (GLOBALVARS NC.OrigReadTable))
(PRINT LinkLabels Stream NC.OrigReadTable])

(**NC.WriteBrowserFormat**
[LAMBDA (Stream Format) (* rht%: " 1-Nov-86 15:37")

(* * Write browser format to notefile.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

(**DECLARE** (GLOBALVARS NC.OrigReadTable))
(PRINT Format Stream NC.OrigReadTable])

(**NC.WriteSpecialBrowserSpecs**
[LAMBDA (Stream LinkLabels) (* rht%: " 1-Nov-86 15:38")

(* * Write special browser specs to notefile.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

(**DECLARE** (GLOBALVARS NC.OrigReadTable))
(PRINT LinkLabels Stream NC.OrigReadTable])

(**NC.WriteBrowserDepth**
[LAMBDA (Stream Depth) (* rht%: " 1-Nov-86 15:38")

(* * Write depth to notefile.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

(**DECLARE** (GLOBALVARS NC.OrigReadTable))
(PRINT Depth Stream NC.OrigReadTable])

(**NC.WriteBrowserSavedLinkingInfo**
[LAMBDA (Stream BrowserSavedLinkingInfo) (* rht%: " 7-Feb-86 00:29")

(* * Write lists of info on multiple links between nodes. Each list consists of a UID cons'ed to a prop list of destination UIDs
and link dashing info.)

(PRIN1 '%(Stream)
(**for** BrowserSavedLinkingInfoForNode **in** BrowserSavedLinkingInfo **eachtime** (BLOCK) **do** (
NC.WriteBrowserSavedLinkingInfoForNode
Stream
BrowserSavedLinkingInfoForNode
))
(PRIN1 '%) Stream])

(**NC.WriteBrowserSavedLinkingInfoForNode**
[LAMBDA (Stream BrowserSavedLinkingInfoForNode) (* rht%: " 1-Nov-86 15:39")

(* * Write list of info on multiple links for a particular node. In form of a source UID cons'ed to a prop list of destination UIDs
and link dashing info.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

(**DECLARE** (GLOBALVARS NC.OrigReadTable))

```
    (PRIN1 '%( Stream)
    (NC.WriteUID Stream (CAR BrowserSavedLinkingInfoForNode))
    (for SavedLinkInfo on (CDR BrowserSavedLinkingInfoForNode) by (CDDR SavedLinkInfo) eachtime (BLOCK)
       do (NC.WriteUID Stream (CAR SavedLinkInfo))
          (PRINT (CADR SavedLinkInfo)
                 Stream NC.OrigReadTable))
    (PRIN1 '%) Stream])
```

(NC.WriteBrowserLinksLegend

```
  [LAMBDA (Stream LinksLegend)                                    (* rht%: " 1-Nov-86 15:38")
```

(* * Write links legend to notefile.)

(* * rht 11/1/86%: Now uses our readtable when printing.)

```
    (DECLARE (GLOBALVARS NC.OrigReadTable))
    (PRINT LinksLegend Stream NC.OrigReadTable])

)

(DEFINEQ
```

(NC.GraphLinkIconUpdateCheck

```
  [LAMBDA (GraphCard Window Graph UpdateIfNullCacheFlg)           ; Edited 28-Jan-88 18:08 by pmi
    ;; Check current values of link icon default display global params against values cached on Window.  If changed, then fix link icon sizes in graph.
    ;; Return non-nil if we had to fix graph nodes.

    ;; rht 2/20/86: Changed to use card's user data area instead of props.

    ;; pmi 1/28/88: Added more global params (NC.LinkIconFont NC.LinkIconMultiLineMode NC.LinkIconMaxWidth and NC.LinkIconBorderWidth) to
    ;; be cached and checked.
    (DECLARE (GLOBALVARS NC.LinkIconShowTitleFlg NC.LinkIconShowLinkTypeFlg NC.LinkIconAttachBitmapFlg
                     NC.LinkIconFont NC.LinkIconMultiLineMode NC.LinkIconMaxWidth NC.LinkIconBorderWidth))
    (LET ((OldGlobalParams (NC.FetchUserDataProp GraphCard 'CachedGlobalParams))
          (CurGlobalParams (LIST NC.LinkIconShowTitleFlg NC.LinkIconShowLinkTypeFlg NC.LinkIconAttachBitmapFlg
                                 NC.LinkIconFont NC.LinkIconMultiLineMode NC.LinkIconMaxWidth
                                 NC.LinkIconBorderWidth))
          DidWorkFlg)
         (if (NOT (EQUAL OldGlobalParams CurGlobalParams))
             then (if (OR OldGlobalParams UpdateIfNullCacheFlg)
                      then (for Node in (fetch (GRAPH GRAPHNODES) of Graph) do (NC.GraphNodeLinkIconUpdate Window
                                                                                   Node))
                           (SETQ DidWorkFlg T))
                  (NC.SetUserDataProp GraphCard 'CachedGlobalParams CurGlobalParams))
         DidWorkFlg])
```

(NC.BrowserRepaintFn

```
  [LAMBDA (Window Region)                                         (* rht%: "15-Nov-85 16:05")
```

(* * Check if cached global params have changed. If so, fix graph nodes before redisplaying.)

```
    (NC.GraphLinkIconUpdateCheck (NC.CoerceToCard Window)
           Window
           (WINDOWPROP Window 'GRAPH)
           NIL])
```

(NC.GetBrowserNodeID

```
  [LAMBDA (BrowserCard NodeCard)                                  (* DSJ%: "11-Nov-87 20:55")
```

(* * Create a browser node atom from a new UID. Hang the card object off as a property for those who need it.)

(* * rht 11/18/85%: Now checks to see if NodeCard already appears in graph before creating a new GraphNodeID.
Use the browser hash array to do the lookup.)

(* * rht 6/1/87%: Changed hash array to map from card UIDs to graph node ids rather than from card objects to graph node ids.)

(* * dsj. |11/12/87.| Changed so won't break if NodeCard if NIL.
It may be in certain 1.2 --> 1.3 conversion cases.)

```
    (LET ((HashArray (NC.HashArrayFromBrowserCard BrowserCard))
          (NodeCardUID (AND NodeCard (fetch (Card UID) of NodeCard)))
          NewUID)
         (if (GETHASH NodeCardUID HashArray)
           else (NC.GraphNodeIDPutProp (SETQ NewUID (NC.MakeBrowserNodeUID))
                       'CardObject NodeCard)
                (PUTHASH NodeCardUID NewUID HashArray)
                NewUID])
```

(NC.MakeBrowserNodeUID

```
  [LAMBDA NIL                                                     (* rht%: "18-Nov-85 21:29")
```

(* * Create a standard UID, but make sure it's an atom.)

```
    (NC.MakeUID])
```

(NC.GetBrowserHashArray

```
  [LAMBDA (BrowserCard Graph)                                    (* rht%: "10-Jun-87 22:41")
```

(* * Build and install a hash array mapping cards to browsernode UIDs, unless one's already there.
If Graph argument is nil, then make a new hash array smashing any existin one.)

(* * rht 4/30/86%: Now makes sure we're not working with a label node instead of a card node.)

(* * rht 6/1/87%: Changed hash array to map from card UIDs to graph node ids rather than from card objects to graph node ids.)

(* * rht 6/10/87%: Now checks that CardObject on GraphNodeID is valid card before stashing in hash array.)

```
    (DECLARE (GLOBALVARS NC.BrowserHashArraySize))
    (if (AND Graph (NC.HashArrayFromBrowserCard BrowserCard))
      else (LET ((HashArray (NC.CreateUIDHashArray NC.BrowserHashArraySize)))
                (NC.SetUserDataProp BrowserCard 'BrowserHashArray HashArray)
                (AND Graph (for GraphNode in (fetch (GRAPH GRAPHNODES) of Graph) bind GraphNodeID
                              eachtime (BLOCK) when (SETQ GraphNodeID (NC.CoerceToGraphNodeID GraphNode))
                              do (LET [(CardObject (NC.GraphNodeIDGetProp GraphNodeID 'CardObject]
                                       (if (NC.ValidCardP CardObject)
                                           then (PUTHASH (fetch (Card UID) of CardObject)
                                                         GraphNodeID HashArray])
```

(NC.RemoveBrowserNodeHashArrayEntry

```
  [LAMBDA (BrowserCard NodeCard)                                 (* rht%: " 1-Jun-87 21:40")
```

(* * Remove the entry for given card from given browser's hash array.)

(* * rht 6/1/87%: Now expects hash array to map from card UIDs to graph node ids rather than from card objects to graph node ids.)

```
    (LET ((HashArray (NC.HashArrayFromBrowserCard BrowserCard)))
         (if HashArray
             then (PUTHASH (fetch (Card UID) of NodeCard)
                           NIL HashArray])
```

(NC.HashArrayFromBrowserCard

```
  [LAMBDA (BrowserCard)                                          (* rht%: "18-Nov-85 22:39")
```

(* * Return the browser hash array for this browser.)

```
    (NC.FetchUserDataProp BrowserCard 'BrowserHashArray])
```

(NC.CardFromBrowserNode

```
  [LAMBDA (GraphNode)                                            ; Edited 21-Jan-88 22:37 by Trigg
    ;; If this is a graph node that corresponds to a valid notecard, then return the card.
    (LET [(Card (NC.CardFromBrowserNodeID (NC.CoerceToGraphNodeID GraphNode]
         (if (NCP.ValidCardP Card)
             then Card])
```

(NC.PutBrowserSubstance

```
  [LAMBDA (Card Stream)                                          (* rht%: " 4-May-87 20:14")
```

(* * For each BrowserUID, clear its UID prop list. Otherwise HPRINT will die in PutGraphSubstance.)

(* * rht 1/23/86%: Now takes Stream as arg)

(* * rht 2/1/86%: Now saves old UID information on card's proplist for restoring when card is brought up again.
Note that this info will only exist for nodes connected to some other node with multiple links.)

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

(* * rht 2/6/86%: Now writes down all browser info to substance rather than letting it live on prop list.)

(* * rht 2/20/86%: Now checks to see if saved linking info is cached before recomputing it.)

(* * fgh |5/25/86| Added default depth paramter.)

(* * rht 5/4/87%: Changed call from NC.ApplySupersFn to APPLY* because otherwise we generate stack overflow for cards that specialize browser card type.)

```
    (NC.WriteBrowserRootsInfo Stream (NC.FetchBrowserRootsInfo Card))
    (NC.WriteBrowserLinkLabels Stream (NC.FetchBrowserLinkLabels Card))
    (NC.WriteBrowserFormat Stream (NC.FetchBrowserFormat Card))
    (NC.WriteSpecialBrowserSpecs Stream (NC.FetchSpecialBrowserSpecs Card))
    (NC.WriteBrowserDepth Stream (OR (NC.FetchBrowserDepth Card)
                                     0))
```

```
    (NC.WriteBrowserLinksLegend Stream (NC.FetchBrowserLinksLegend Card))
    [NC.WriteBrowserSavedLinkingInfo Stream (OR (NC.FetchBrowserSavedLinkingInfo Card)
                                                (NC.SetBrowserSavedLinkingInfo Card (NC.ComputeBrowserSavedLinkingInfo
                                                                                     Card]
    (APPLY* (NCP.CardTypeFn 'Graph 'PutFn)
           Card Stream])
```

(NC.FetchBrowserRoots

```
  [LAMBDA (Card)                                                    (* rht%: " 7-Feb-86 12:29")

          (* * Get roots off Card's UID list and convert them from UID pairs to Card objects.)

    (for BrowserRootUIDPair in (NC.FetchBrowserRootsInfo Card) eachtime (BLOCK)
       collect (NC.CardFromUID (CAR BrowserRootUIDPair)
                     (NC.NoteFileFromNoteFileUID (CDR BrowserRootUIDPair])
```

(NC.SetBrowserRoots

```
  [LAMBDA (Card BrowserRoots)                                       (* rht%: " 6-Feb-87 18:38")

          (* * Put the roots on the Card's UID property list in the CardUID/NoteFileUID pair format.)

          (* * rht 2/6/87%: Now checks that BrowserRoots are valid cards.)

    (NC.SetBrowserRootsInfo Card (for BrowserRoot in BrowserRoots eachtime (BLOCK) when (NC.ValidCardP BrowserRoot)
                                  collect (CONS (fetch (Card UID) of BrowserRoot)
                                                (fetch (NoteFile UID) of (fetch (Card NoteFile) of BrowserRoot])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS NC.ArrowHeadLength NC.ArrowHeadAngle NC.ArrowHeadXVal NC.ArrowHeadYVal)
)

(RPAQ? NC.ArrowHeadLength 7)

(RPAQ? NC.ArrowHeadAngle 20)

(RPAQ? NC.ArrowHeadXVal (TIMES NC.ArrowHeadLength (COS NC.ArrowHeadAngle)))

(RPAQ? NC.ArrowHeadYVal (TIMES NC.ArrowHeadLength (SIN NC.ArrowHeadAngle)))
```

;;; init

```
(DEFINEQ
```

(NC.AddBrowserCard

```
  [LAMBDA NIL                                                       ; Edited  2-Dec-88 17:01 by krivacic
    ;; fgh 11/14/85 Updated to handle merge of card and substance types.
    ;; rht 4/7/86: Added middle button menu items.
    ;; rht 4/25/87: Added QuitFn
    ;; rht 7/29/88: Added ChangeBrowserRoots item to middle button menu
    (DECLARE (GLOBALVARS NC.GlobalInsertLinkMenuItem))
    (NC.AddCardType 'Browser 'Graph `[(MakeFn ,(FUNCTION NC.MakeBrowserCard))
                                     (EditFn ,(FUNCTION NC.BringUpBrowserCard))
                                     (PutFn ,(FUNCTION NC.PutBrowserSubstance))
                                     (GetFn ,(FUNCTION NC.GetBrowserSubstance))
                                     (DeleteLinksFn ,(FUNCTION NC.DelReferencesToCardFromBrowser))
                                     (QuitFn ,(FUNCTION NC.BrowserCardQuitFn]
           `((LinkDisplayMode Title)
             (DefaultHeight 350)
             (DefaultWidth 500)
             (DisplayedInMenuFlg ,T)
             (LinkIconAttachedBitMap ,NC.BrowserCardIcon)
             [LeftButtonMenuItems ,(for Item in (NC.GetCardTypeField LeftButtonMenuItems 'Graph)
                                      collect (if (EQ (CAR Item)
                                                      'Insert% Link)
                                                  then NC.GlobalInsertLinkMenuItem
                                                  else Item]
             (MiddleButtonMenuItems ,'((Recompute% Browser (FUNCTION NC.UpdateBrowserCard)
                                                  "Recomputes this browser to show the current state of the
                                                  NoteFile.")
                                       (Relayout% Graph (FUNCTION NC.RelayoutBrowserCard)
                                                  "Re-layout the browser, but keep same nodes.")
                                       (Reconnect% Nodes (FUNCTION NC.ConnectNodesInBrowser)
                                                  "Draw all possible links, from currently selected link types,
                                                  between pairs of nodes.")
                                       (Unconnect% Nodes (FUNCTION NC.UnconnectNodesInBrowser)
                                                  "Undraw all links in the browser.")
                                       (|Expand Browser Node| (FUNCTION NC.ExpandBrowserNode)
                                                  "Expand the graph under one node to a given depth.")
                                       (|Graph Edit Menu| (FUNCTION NC.GetGraphEditMenu)
```

```
                                        "Bring up the graph editor menu.")
                                  (|Change Browser Specs| (FUNCTION NC.ChangeBrowserSpecs)
                                        "Make changes to some or all of the browser specs, e.g. link
                                        types, depth, etc.")
                                  (|Change Browser Roots| (FUNCTION NC.ChangeBrowserRoots)
                                        "Change which nodes in the browser are considered roots.")
                                  (|Browser Overview Win| (FUNCTION NC.MakeBrowserOverviewWin)
                                        "Attach the browser overview window.")
                                  (|Change Overview Specs| (FUNCTION NC.AskBrowserOverviewSpecs)
                                        "Change the browser overview specs: where to attach and what
                                        mode."])

)

(RPAQQ NC.BrowserCardIcon  )

(DECLARE%: DONTEVAL@LOAD

(NC.AddBrowserCard)
)

(PUTPROPS NCBROWSERCARD FILETYPE :FAKE-COMPILE-FILE)

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

(PUTPROPS NCBROWSERCARD COPYRIGHT ("Venue & Xerox Corporation" 1985 1986 1987 1988 1989 1990 1993 1994 2020)
)
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX