*File created:* 3-Mar-88 13:51:10 **{ERINYES}<LISPUSERS>LYRIC>EQUATIONS.;1**

*changes to:* (FNS EQIO.Put EQIO.Get)

*previous date:* 27-May-87 11:20:49 {IE:PARC:XEROX}<LISP>LYRIC>LISPUSERS>EQUATIONS.;1

*Read Table:* INTERLISP

*Package:* INTERLISP

*Format:* XCCS

```
(RPAQQ EQUATIONSCOMS
       (

;;; EQUATION module: Part 1 of 3

                                                                          ; functions for image object
        (FNS EQIO.CreateFns EQIO.Create EQIO.Imagebox EQIO.Display EQIO.ButtonEventIn EQIO.Copy EQIO.CopyList
             EQIO.Get EQIO.Put EQIO.WhenDeleted EQIO.SelectRegion EQIO.Selection EQIO.DefaultSelectFn
             EQIO.MakeSelectionMenu)


;;; functions to handle individual equation props and data

        (FNS EQIO.EqnType EQIO.EqnDataList EQIO.SetDataList EQIO.EqnData EQIO.EqnProperty EQIO.AllProps
             EQIO.Specify EQIO.GetInitialProps EQIO.NumPieces EQIO.NewStructure)


;;; functions to handle equation specification info

        (FNS EQIO.AddType EQIO.GetInfo EQIO.SetInfo EQIO.TypeProp EQIO.ResetTypeProps EQIO.IsDefined EQIO.GetBox
             EQIO.GetDataSpec EQIO.GetDataSpecList EQIO.GetDataPosition EQIO.GetDataSelectRegion EQIO.MakeSpec
             EQIO.MakeDataSpec)


;;; variable specification

        (GLOBALVARS EquationInfo EquationTypeMenu EquationImageFns UnknownEquationData EquationDefaultSelectFn)
        (VARS (EquationImageFns NIL))
        (INITVARS EquationInfo (EquationDefaultSelectFn 'EQIO.DefaultSelectFn))
        [P (TEDIT.ADD.MENUITEM TEDIT.DEFAULT.MENU '(Equation 'EQN.Equation]
        (P                                                                ; needed to force the getfn to be recognized before any new
                                                                          ; eqns defined
           (SETQ EquationImageFns (EQIO.CreateFns)))
        (VARS UnknownEquationData)
        (PROP ARGNAMES EQIO.TypeProp EQIO.NumPieces EQIO.AllProps EQIO.EqnProperty)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA)
                      (NLAML)
                      (LAMA EQN.ObjEditWindow EQN.ContinueFlg EQN.PieceNumber EQN.ResultObj EQN.ResultWindow
                            EQN.EditWindow EQIO.TypeProp EQIO.NumPieces EQIO.AllProps EQIO.EqnProperty)))


;;; EQUATIONEDIT module: Part 2 of 3

                                                                          ; functions to edit data pieces
        (FNS EQN.AbortEdit EQN.StopEdit EQN.ContinueEdit EQN.FinishEdit EQN.MakeEditWindow EQN.SetUpEdit
             EQN.StartEdit EQN.StartNextEdit EQN.UpdateEdit EQN.DefaultData EQN.TypeMenu)


;;; hooks to control behavior of equation subeditor

        (FNS EQN.Equation EQN.NextPiece EQN.FinishEqn EQN.NoUpdateAbort EQN.PreventUpdate EQN.CharFn
             EQN.TEditSpecialChar EQN.SnuggleWindows EQN.SnuggleMainWindow)


;;; functions to handle equation fonts

        (FNS EQN.EquationFontNumber EQN.EquationFont EQN.GetEqnFont EQN.MakeFS)


;;; utilities

        (FNS EQN.AdjustWindow EQN.CheckWindowSize)
        (FNS EQN.SubEditorP EQN.WindowFromText EQN.EditWindow EQN.ResultWindow EQN.ResultObj EQN.PieceNumber
             EQN.ContinueFlg EQN.ValidEditWindow EQN.ObjEditWindow)
        (FNS EQN.Make)
        (GLOBALVARS EquationFontSpecs)
        (VARS EquationFontSpecs)
        (PROP ARGNAMES EQN.ObjEditWindow EQN.ContinueFlg EQN.PieceNumber EQN.ResultObj EQN.ResultWindow
              EQN.EditWindow)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS
               (ADDVARS (NLAMA)
                      (NLAML)
                      (LAMA EQN.ObjEditWindow EQN.ContinueFlg EQN.PieceNumber EQN.ResultObj EQN.ResultWindow
                            EQN.EditWindow)))
```

;;; FORMATSTRING module: Part 3 of 3

```
        (FNS FS.Box FS.Copy FS.Display FS.Get FS.Put FS.ItemFont FS.ItemValue FS.ItemShift FS.MakeItem
             FS.Extract FS.ExtractFont FS.ExtractShift FS.Insert FS.AllowedChar FS.RealStringP)
```

;;; Now load EQUATIONFORMS

```
        (FILES EQUATIONFORMS)))
```

;;; EQUATION module: Part 1 of 3

;; functions for image object

```
(DEFINEQ
```

**(EQIO.CreateFns**

```
  [LAMBDA NIL                                                          (* thh%: " 5-Dec-85 10:50")
    (IMAGEFNSCREATE (FUNCTION EQIO.Display)
           (FUNCTION EQIO.Imagebox)
           (FUNCTION EQIO.Put)
           (FUNCTION EQIO.Get)
           (FUNCTION EQIO.Copy)
           (FUNCTION EQIO.ButtonEventIn)
           (FUNCTION NILL)
           (FUNCTION NILL)
           (FUNCTION NILL)
           (FUNCTION EQIO.WhenDeleted)
           (FUNCTION NILL)
           (FUNCTION NILL)
           (FUNCTION NILL])
```

**(EQIO.Create**

```
  [LAMBDA (kind dataList fontSpec objectProps)                         (* thh%: " 5-Dec-85 10:50")
                                                                       (* makes an equation image object of specified kind and data)
                                                                       (* fontSpec, if non-NIL, is used as fontSpec prop of eqn)

             (* objectProps specifies properties for this object -- if NIL then default props for this kind will be used)

    (PROG [(obj (IMAGEOBJCREATE (CONS kind dataList)
                       (COND
                          ((AND (BOUNDP 'EquationImageFns)
                                (IMAGEFNSP EquationImageFns))
                           EquationImageFns)
                          (T (SETQ EquationImageFns (EQIO.CreateFns]
          [COND
             ((NOT objectProps)
              (SETQ objectProps (COPYALL (EQIO.GetInfo kind 'objProps]
          (EQIO.AllProps obj objectProps)
          [COND
             (fontSpec (EQIO.EqnProperty obj 'fontSpec (COND
                                                          ((NUMBERP fontSpec)
                                                           (EQN.GetEqnFont fontSpec))
                                                          (T fontSpec]
          (RETURN obj])
```

**(EQIO.Imagebox**

```
  [LAMBDA (eqnObj imageStream)                                         (* THH "21-May-85 12:13")
                                                                       (* determines size of equation)
    (PROG ((spec (EQIO.Specify eqnObj imageStream)))
          (RETURN (COND
                     (spec (EQIO.GetBox spec))
                     (T (FS.Box UnknownEquationData imageStream])
```

**(EQIO.Display**

```
  [LAMBDA (eqnObj imageStream)                                         (* thh%: "31-May-85 09:30")
                                                                       (* displays equation)
    (PROG [(curX (DSPXPOSITION NIL imageStream))
           (curY (DSPYPOSITION NIL imageStream))
           (spec (EQIO.Specify eqnObj imageStream T))
           (invertPiece (PROG ((editWindow (EQN.ObjEditWindow eqnObj)))
                              (RETURN (COND
                                         (editWindow (EQN.PieceNumber editWindow))
                                         (T NIL]
          (COND
             [spec (for i from 1 to (EQIO.NumPieces eqnObj) bind data pos (desc _ (fetch YDESC of (EQIO.GetBox spec)
                                                                                     ))
                      do (SETQ data (EQIO.EqnData eqnObj i))
```

```
                         (SETQ pos (EQIO.GetDataPosition (EQIO.GetDataSpec spec i)))
                         (COND
                            ((AND pos data)
                             (MOVETO (PLUS curX (fetch XCOORD of pos))
                                    (PLUS curY (fetch YCOORD of pos)
                                          (MINUS desc))
                                    imageStream)
                             (FS.Display data imageStream (EQ invertPiece i]
             (T (FS.Display UnknownEquationData imageStream invertPiece])
```

## (EQIO.ButtonEventIn

```
  [LAMBDA (eqnObj windowStream selection relX relY window textStream button)
                                                                     (* thh%: "19-Mar-86 09:23")
                                                                     (* handles button press in equation)
    (PROG ((editWindow (EQN.EditWindow window))
           [CHOICEMENU (CONSTANT (create MENU
                                         CENTERFLG _ T
                                         ITEMS _ '((Select 'SELECT "Select the image object")
                                                   (Edit 'EDIT "Edit selected piece of the equation"]
           underEdit)
          [SETQ underEdit (AND editWindow (EQ eqnObj (EQN.ResultObj editWindow]
                                                                     (* non-NIL if eqnObj is currently being edited)
          (RETURN (COND
                     ((OR (KEYDOWNP 'RSHIFT)
                          (KEYDOWNP 'LSHIFT)
                          (KEYDOWNP 'CTRL))

          (* note%: using COPYBUTTONEVENTIN fn instead of this test doesnt work since it is never called)

                      (COND
                         (underEdit                                  (* abort sub edit when obj specially selected in main window)
                                (EQN.AbortEdit window)
                                'CHANGED)
                         (T NIL)))
                     (T (SPAWN.MOUSE)
                        (OR (COND
                               [(EQ button 'LEFT)                    (* check for direct selection of piece within the equation)
                                (AND (EQ (MENU CHOICEMENU)
                                         'EDIT)
                                     (LET ((piece# (EQIO.Selection eqnObj windowStream relX relY)))
                                          (COND
                                             (piece# (EQN.StartEdit eqnObj window piece#)
                                                     'CHANGED)
                                             (T NIL]
                               [(EQ button 'MIDDLE)                  (* use menu to allow selection of piece to edit)
                                (LET ((selectFn (EQIO.GetInfo (EQIO.EqnType eqnObj)
                                                        'specialSelectFn))
                                      piece#)
                                     (COND
                                        ((NOT selectFn)
                                         (SETQ selectFn EquationDefaultSelectFn)))
                                     (COND
                                        ((AND selectFn (SETQ piece# (APPLY* selectFn eqnObj)))
                                         (EQN.StartEdit eqnObj window piece#)
                                         'CHANGED)
                                        (T NIL]
                               (T NIL))
                            (LET [(wholeEditFn (EQIO.GetInfo (EQIO.EqnType eqnObj)
                                                      'wholeEditFn]
                                                                     (* treat as top level selection and check for edits of global
                                                                     properties)
                                 (COND
                                    (underEdit (EQN.AbortEdit window)))
                                 (COND
                                    ((AND wholeEditFn (APPLY* wholeEditFn eqnObj window button))
                                     'CHANGED)
                                    (underEdit 'CHANGED)
                                    (T NIL])
```

## (EQIO.Copy

```
  [LAMBDA (eqnObj)                                                   (* THH "12-Jul-85 10:40")
    (LET ((dataList (EQIO.EqnDataList eqnObj)))
         (EQIO.Create (COPY (EQIO.EqnType eqnObj))
                (for data in dataList collect (FS.Copy data))
                NIL
                (EQIO.CopyList (EQIO.AllProps eqnObj])
```

## (EQIO.CopyList

```
  [LAMBDA (list)                                                     (* THH "12-Jul-85 10:44")
                                                                     (* copies list down to atoms or strings but sets anything else to
                                                                     NIL)
                                                                     (* datatype values set to NIL -- allows caching such data only)
    (COND
       ((OR (ATOM list)
            (STRINGP list))
        list)
       ((LISTP list)
        (for item in list collect (EQIO.CopyList item)))
       (T                                                    (* set value to NIL)
          NIL])
```

## (EQIO.Get

```
  [LAMBDA (fileStream)                                        ; Edited  3-Mar-88 13:45 by thh:

    ;; specify readtable for input

    (LET ((*READTABLE* (FIND-READTABLE "INTERLISP")))
         (PROG ((kind (READ fileStream))
                (dataList (FS.Get fileStream)))
               (RETURN (EQIO.Create kind dataList NIL (READ fileStream])
```

## (EQIO.Put

```
  [LAMBDA (eqnObj fileStream)                                 ; Edited  3-Mar-88 13:43 by thh:

    ;; specify readtable for output and eliminate all non-atom/string props which PRIN2 won't handle correctly

    (LET ((*READTABLE* (FIND-READTABLE "INTERLISP")))
         (PRIN2 (EQIO.EqnType eqnObj)
                fileStream)
         (FS.Put (EQIO.EqnDataList eqnObj)
                fileStream)
         (PRIN2 (EQIO.CopyList (EQIO.AllProps eqnObj))
                fileStream])
```

## (EQIO.WhenDeleted

```
  [LAMBDA (eqnObj window)                                     (* thh%: "15-May-85 11:27")
                                                              (* called when eqnObj is about to be deleted from edit window)
                                                              (* abort any sub edit of this object)
    (PROG ((editWindow (EQN.ValidEditWindow (EQN.EditWindow window)
                               eqnObj)))
          (COND
             (editWindow (EQN.AbortEdit window])
```

## (EQIO.SelectRegion

```
  [LAMBDA (spec data piece# imageStream)                      (* THH "21-May-85 12:13")
                                                              (* determines selection region for piece in eqnObj)
    (PROG ((dataSpec (EQIO.GetDataSpec spec piece#)))
          (RETURN (COND
                     ((EQIO.GetDataSelectRegion dataSpec))
                     [data (PROG ((dataBox (FS.Box data imageStream))
                                  (pos (EQIO.GetDataPosition dataSpec)))
                                 (RETURN (create REGION
                                                LEFT _ (fetch XCOORD of pos)
                                                BOTTOM _ (DIFFERENCE (fetch YCOORD of pos)
                                                                (fetch YDESC of dataBox))
                                                WIDTH _ (fetch XSIZE of dataBox)
                                                HEIGHT _ (fetch YSIZE of dataBox]
                     (T NIL])
```

## (EQIO.Selection

```
  [LAMBDA (eqnObj imageStream relX relY)                      (* thh%: "31-May-85 09:31")
                                                              (* returns piece number of data within which selection was
                                                              made, if any)

          (* note%: if slection region for a piece is zero size, then cannot select that piece)

          (* new TEdit%: relY is measured from baseline of object so must adjust since eqn forms measure regions w.r.t.
          l.l. corner of box)

    (PROG ((spec (EQIO.Specify eqnObj imageStream))
           piece#)
          (RETURN (COND
                     [spec (add relY (fetch YDESC of (EQIO.GetBox spec)))
                           (SETQ piece# (for i from 1 to (EQIO.NumPieces eqnObj) bind region
                                           do (SETQ region (EQIO.SelectRegion spec (EQIO.EqnData eqnObj i)
                                                                  i imageStream))
                                              (COND
                                                 ((AND region (INSIDEP region relX relY))
                                                  (RETURN i]
                     (T                                       (* unknown equation -- not able to select)
                        NIL])
```

## (EQIO.DefaultSelectFn

```
  [LAMBDA (eqnObj)                                            (* thh%: "31-May-85 08:34")

          (* provides a menu-based selection of a part of the equation -- returns the piece# of the part selected or NIL if no part was
          selected)
```

```
              (* this is useful for selecting parts whose select region on the screen is zero size)

    (PROG ((type (EQIO.EqnType eqnObj))
           menu)
          [COND
             [(EQIO.GetInfo type 'variable?)
              (SETQ menu (EQIO.EqnProperty eqnObj 'selectionMenu))
              (COND
                 ((NOT (type? MENU menu))
                  (SETQ menu (EQIO.MakeSelectionMenu type (EQIO.NumPieces eqnObj)))
                  (EQIO.EqnProperty eqnObj 'selectionMenu menu]
             (T (SETQ menu (EQIO.TypeProp type 'selectionMenu))
                (COND
                   ((NOT (type? MENU menu))
                    (SETQ menu (EQIO.MakeSelectionMenu type (EQIO.NumPieces eqnObj)))
                    (EQIO.TypeProp type 'selectionMenu menu]
          (RETURN (COND
                     (menu (MENU menu))
                     (T NIL])
```

## (EQIO.MakeSelectionMenu

```
  [LAMBDA (type numPieces)                                    (* thh%: "19-Mar-86 09:30")
                                                              (* creates a selection menu for the specified type of equation)
                                                              (* numPieces may be NIL if this is an unknown type of equation)

    (COND
       [(AND (FIXP numPieces)
             (IGREATERP numPieces 0))
        (PROG [(pieceNames (EQIO.GetInfo type 'pieceNames]
              (RETURN (create MENU
                              CENTERFLG _ T
                              TITLE _ "Eqn piece?"
                              ITEMS _ (for i from 1 to numPieces bind name collect (SETQ name (CAR pieceNames))
                                                                                  (COND
                                                                                     (name (SETQ pieceNames
                                                                                            (CDR pieceNames))
                                                                                           (LIST name i))
                                                                                     (T i]
       (T NIL])
)
```

;;; functions to handle individual equation props and data

```
(DEFINEQ
```

## (EQIO.EqnType

```
  [LAMBDA (eqnObj)                                            (* THH " 2-May-85 12:59")
                                                              (* returns type of equation)

    (CAR (IMAGEOBJPROP eqnObj 'OBJECTDATUM])
```

## (EQIO.EqnDataList

```
  [LAMBDA (eqnObj)                                            (* THH " 2-May-85 12:59")
                                                              (* returns list of data pieces in the equation)

    (CDR (IMAGEOBJPROP eqnObj 'OBJECTDATUM])
```

## (EQIO.SetDataList

```
  [LAMBDA (eqnObj newDataList)                                (* thh%: " 3-Jun-85 08:34")

              (* replaces entire data list of eqn -- caller must make sure any props, e.g.
              numPieces, are suitably adjusted)

    (IMAGEOBJPROP eqnObj 'OBJECTDATUM (CONS (EQIO.EqnType eqnObj)
                                            newDataList])
```

## (EQIO.EqnData

```
  [LAMBDA (eqnObj piece#)                                     (* THH " 2-May-85 13:52")
    (CAR (NTH (EQIO.EqnDataList eqnObj)
              piece#])
```

## (EQIO.EqnProperty

```
  [LAMBDA eqn                                                 (* THH " 8-May-85 08:48")
                                                              (* gets and sets individual eqn props)
                                                              (* eqn is of form (eqnObj prop {newValue}))

    (COND
       ((IEQP eqn 2)
        (LISTGET (IMAGEOBJPROP (ARG eqn 1)
                        'props)
               (ARG eqn 2)))
       ((IEQP eqn 3)
        (PROG [(props (IMAGEOBJPROP (ARG eqn 1)
                         'props]
              [COND
                 (props (LISTPUT props (ARG eqn 2)
                                (ARG eqn 3)))
                 (T (SETQ props (LIST (ARG eqn 2)
                                      (ARG eqn 3]
              (IMAGEOBJPROP (ARG eqn 1)
                     'props props])
```

(**EQIO.AllProps**

```
  [LAMBDA eqn                                                    (* THH " 8-May-85 08:48")
                                                                 (* gets and sets all props for eqnObj)
                                                                 (* eqn is of form (eqnObj {newProps}))
    (COND
       ((IEQP eqn 1)
        (IMAGEOBJPROP (ARG eqn 1)
               'props))
       ((IEQP eqn 2)
        (IMAGEOBJPROP (ARG eqn 1)
               'props
               (ARG eqn 2])
```

(**EQIO.Specify**

```
  [LAMBDA (eqnObj imageStream draw?)                             (* THH " 2-May-85 13:45")

          (* returns specification for equation on imageStream, and if draw? is not NIL, draws the non-data parts of the equation)

    (PROG [(formFn (EQIO.GetInfo (EQIO.EqnType eqnObj)
                          'formFn]
          (RETURN (COND
                     (formFn (APPLY* formFn eqnObj imageStream draw?))
                     (T NIL])
```

(**EQIO.GetInitialProps**

```
  [LAMBDA (type)                                                 (* thh%: "31-May-85 09:02")
                                                                 (* gets initial prop list to use when new equation of specified
                                                                 type is created)
    (PROG ([props (COPY (EQIO.GetInfo type 'objectProps]
           (initialPropFn (EQIO.GetInfo type 'initialPropFn))
           newProps)
          [COND
             ((AND initialPropFn (SETQ newProps (APPLY* initialPropFn type)))
              (COND
                 [props (repeatwhile newProps do (LISTPUT props (CAR newProps)
                                                           (CADR newProps))
                                                   (SETQ newProps (CDDR newProps]
                 (T (SETQ props newProps]
          (COND
             ([AND props (LISTGET props 'numPieces)
                   (NOT (EQIO.GetInfo type 'variable?]           (* this equation does not allow a variable number of pieces)
              (ERROR "EQIO.GetInitialProps: can't specify numPieces for fixed size eqn type = " type)))
          (RETURN props])
```

(**EQIO.NumPieces**

```
  [LAMBDA eqn                                                    (* thh%: "31-Jul-85 08:49")
                                                                 (* gets or sets current number of parts for eqn --
                                                                 args are (eqnObj {newValue}))
    (COND
       ((IGREATERP eqn 0)
        (PROG ((eqnObj (ARG eqn 1))
               type value)
              (SETQ type (EQIO.EqnType eqnObj))
              (RETURN (COND
                         [(IEQP eqn 1)
                          (COND
                             ([AND (EQIO.GetInfo type 'variable?)
                                   (FIXP (SETQ value (EQIO.EqnProperty eqnObj 'numPieces]
                              value)
                             (T                                  (* not variable or different number of parts not specified)
                                (EQIO.GetInfo type 'numPieces]
                         ((EQIO.GetInfo type 'variable?)
                          (EQIO.EqnProperty eqnObj 'numPieces (ARG eqn 2))
                          (EQIO.NewStructure eqnObj))
                         (T (ERROR "EQIO.NumPieces: equation has fixed # of parts, type = " type])
```

(**EQIO.NewStructure**

```
  [LAMBDA (eqnObj)                                               (* thh%: " 3-Jun-85 09:21")

          (* called when eqn structure is changed -- e.g. different number of parts --
          to reset any saved menus, etc.)

    (EQIO.EqnProperty eqnObj 'selectionMenu NIL)
    (PROG [(changeFn (EQIO.GetInfo (EQIO.EqnType eqnObj)
                            'changeFn]
          (COND
             (changeFn (APPLY* changeFn eqnObj])

)
```

;;; functions to handle equation specification info

```
(DEFINEQ

(EQIO.AddType
  [LAMBDA (type formFn numPieces PROPS)                        (* THH " 1-Jul-85 08:37")
                                                                (* creates info for new equation type)
    [PROG ((newValue (APPEND (LIST 'formFn formFn 'numPieces numPieces)
                            PROPS)))
          (PUTPROP type 'equationInfo newValue)
          (COND
             ((NOT (MEMB type EquationInfo))
              (push EquationInfo type]
    (EQIO.ResetTypeProps type])

(EQIO.GetInfo
  [LAMBDA (type info)                                          (* thh%: "28-Jun-85 15:17")
                                                                (* returns specified info for equation type)
    (LISTGET (GETPROP type 'equationInfo)
           info])

(EQIO.SetInfo
  [LAMBDA (type info newValue)                                 (* thh%: "28-Jun-85 15:38")
                                                                (* allows setting particular equation info items for previously
                                                                defined equation types)
    (COND
       ((EQIO.IsDefined type)
        (LET [(spec (GETPROP type 'equationInfo]
             (COND
                (spec (LISTPUT spec info newValue)
                      (PUTPROP type 'equationInfo spec)
                      (EQIO.ResetTypeProps type)
                      newValue)
                (T (ERROR "EQIO.SetInfo: warning -- no specifications for eqn type = " type])

(EQIO.TypeProp
  [LAMBDA type                                                 (* thh%: "31-May-85 09:11")

          (* associates properties with equation types -- e.g. can be used to store selection menus for equations with fixed number of
          parts)
                                                                (* args are (type prop {newValue}))
    (COND
       ((IEQP type 2)
        (LISTGET (GETPROP (ARG type 1)
                        'equationProps)
               (ARG type 2)))
       ((IEQP type 3)
        (PROG [(list (GETPROP (ARG type 1)
                            'equationProps]
              [COND
                 (list (LISTPUT list (ARG type 2)
                              (ARG type 3)))
                 (T (SETQ list (LIST (ARG type 2)
                                     (ARG type 3]
              (PUTPROP (ARG type 1)
                     'equationProps list])

(EQIO.ResetTypeProps
  [LAMBDA (type)                                               (* THH " 1-Jul-85 08:36")
                                                                (* removes all props associated with this equation type --
                                                                called when type info redefined)
    (SETQ EquationTypeMenu NIL)
    (PUTPROP type 'equationProps NIL])

(EQIO.IsDefined
  [LAMBDA (type)                                               (* thh%: "28-Jun-85 15:27")
                                                                (* returns type if it is a currently defined equation type, else NIL)
    (COND
       ((MEMB type EquationInfo)
        type)
       (T NIL])

(EQIO.GetBox
  [LAMBDA (specification)                                  (* THH " 2-May-85 13:22")
                                                           (* gets image box)
    (CAR specification])

(EQIO.GetDataSpec
  [LAMBDA (specification piece#)                           (* THH " 2-May-85 13:26")
                                                           (* gets data spec for corresponding piece)
    (CAR (NTH specification (ADD1 piece#])

(EQIO.GetDataSpecList
  [LAMBDA (specification)                                  (* THH "12-Jul-85 10:22")
    (CDR specification])

(EQIO.GetDataPosition
  [LAMBDA (dataSpec)                                       (* THH " 2-May-85 13:26")
    (CAR dataSpec])

(EQIO.GetDataSelectRegion
  [LAMBDA (dataSpec)                                       (* THH " 2-May-85 13:26")
    (CDR dataSpec])

(EQIO.MakeSpec
  [LAMBDA (box dataSpecList)                               (* THH " 2-May-85 13:26")
                                                           (* constructs specification)
    (CONS box dataSpecList])

(EQIO.MakeDataSpec
  [LAMBDA (position selectRegion)                          (* THH " 2-May-85 13:26")
    (CONS position selectRegion])
)

;;; variable specification

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS EquationInfo EquationTypeMenu EquationImageFns UnknownEquationData EquationDefaultSelectFn)
)

(RPAQQ EquationImageFns NIL)

(RPAQ? EquationInfo NIL)

(RPAQ? EquationDefaultSelectFn 'EQIO.DefaultSelectFn)

[TEDIT.ADD.MENUITEM TEDIT.DEFAULT.MENU '(Equation 'EQN.Equation]

;; needed to force the getfn to be recognized before any new eqns defined

(SETQ EquationImageFns (EQIO.CreateFns))

(RPAQQ UnknownEquationData (((Gacha 10)
                             "[unknown equation]")))

(PUTPROPS EQIO.TypeProp ARGNAMES (NIL (type prop {newValue})
                                      args))

(PUTPROPS EQIO.NumPieces ARGNAMES (NIL (eqnObj {newValue})
                                       args))

(PUTPROPS EQIO.AllProps ARGNAMES (NIL (eqnObj {newValue})
                                      args))

(PUTPROPS EQIO.EqnProperty ARGNAMES (NIL (eqnObj prop {newValue})
                                         args))

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA EQN.ObjEditWindow EQN.ContinueFlg EQN.PieceNumber EQN.ResultObj EQN.ResultWindow EQN.EditWindow
                     EQIO.TypeProp EQIO.NumPieces EQIO.AllProps EQIO.EqnProperty)
)

;;; EQUATIONEDIT module: Part 2 of 3

;; functions to edit data pieces
```

```
(DEFINEQ

(EQN.AbortEdit
  [LAMBDA (mainWindow)                                                    (* thh%: " 3-May-85 11:17")
                                                                          (* terminates any eqn edit without allowing any continuation)
    (PROG ((editWindow (EQN.EditWindow mainWindow)))
          (COND
             ((WINDOWP editWindow)
              (EQN.ContinueFlg editWindow NIL)
              (EQN.StopEdit mainWindow])

(EQN.StopEdit
  [LAMBDA (mainWindow)                                                    (* thh%: " 3-May-85 10:31")
                                                                          (* terminates any eqn edit)
    (PROG ((editWindow (EQN.EditWindow mainWindow)))
          (COND
             ((WINDOWP editWindow)
              (EQN.FinishEdit editWindow)
              (TEDIT.QUIT (TEXTSTREAM editWindow))
              (CLOSEW editWindow])

(EQN.ContinueEdit
  [LAMBDA (window)                                                        (* thh%: "28-Jun-85 14:14")

          (* called after sub edit quits to see if eqn edit should continue with next piece)

          (* continueFlg is NIL to stop, T to continue incrementing by 1, FIXP to increment by that amount once,
          (FIXP . end) to increment up/down to end (end = NIL to contnue until no more)%, or a function to determine next piece from
          current piece (NIL to end))
                                                                          (* selection is pending delete for selections that continue beyond
    current piece)
    [PROG [(process (WINDOWPROP (EQN.ResultWindow window)
                           'PROCESS]                                      (* return TTY process to result window)
          (COND
             (process (TTY.PROCESS process]
    (PROG ((continueFlg (EQN.ContinueFlg window))
           (newPiece# (EQN.PieceNumber window))
           increment end)
          (COND
             ((COND
                 ((EQ continueFlg T)
                  (add newPiece# 1)
                  T)
                 ((NULL continueFlg)
                  NIL)
                 [(FIXP continueFlg)                                      (* treat pieces as circular list)
                  (LET [(oldPiece# newPiece#)
                        (numPieces (EQIO.NumPieces (EQN.ResultObj window]
                       (SETQ newPiece# (IMOD (PLUS oldPiece# continueFlg)
                                             numPieces))
                       (COND
                          ((ZEROP newPiece#)
                           (SETQ newPiece# numPieces)))
                       (SETQ continueFlg NIL)                             (* this is a one-shot continuation so continueFlg for new edit to
                                                                          NIL)
                       (NOT (IEQP newPiece# oldPiece#]
                 ((LISTP continueFlg)
                  (COND
                     [(AND (FIXP (SETQ increment (CAR continueFlg)))
                           (NOT (ZEROP increment))
                           (OR (FIXP (SETQ end (CDR continueFlg)))
                               (NULL end)))
                      (add newPiece# increment)
                      (COND
                         ((NULL end)                                      (* always continue)
                          T)
                         ((MINUSP increment)
                          (IGEQ newPiece# end))
                         (T (ILEQ newPiece# end]
                     (T                                                   (* bad format list -- don't continue)
                        NIL)))
                 ((LITATOM continueFlg)
                  (SETQ newPiece# (APPLY* continueFlg newPiece#))
                                                                          (* continue if the function returned a valid piece number)
                  (FIXP newPiece#))
                 (T (ERROR "EQN.ContinueEdit: Invalid value for continueFlg = " continueFlg)
                    NIL))                                                 (* continue with edit of newPiece#)

          (* * continuation of edit requested -- start new edit)

              (EQN.StartNextEdit (EQN.ResultObj window)
                     (EQN.ResultWindow window)
                     newPiece# continueFlg (COND
                                              (continueFlg 'PENDINGDEL)
                                             (T NIL])

(EQN.FinishEdit
  [LAMBDA (window)                                          (* thh%: " 5-Dec-85 11:06")
                                                            (* returns edited data to calling TEdit)
    (PROG ((mainWindow (EQN.ResultWindow window)))
          (EQN.ObjEditWindow (EQN.ResultObj window)
                 NIL)
          (EQN.AbortEdit window)                            (* abort any sub edits)
          (EQN.UpdateEdit mainWindow (WINDOWPROP window 'NOUPDATE))
          (EQN.EditWindow mainWindow NIL))                  (* return T to allow TEdit to quit without user confirmation)
    T])

(EQN.MakeEditWindow
  [LAMBDA (mainWindow XSIZE YSIZE)                          (* thh%: " 6-Dec-85 11:34")
                                                            (* creates window for subeditor)
    (LET ((editWindow (CREATEW (CREATEREGION 0 0 (fetch (REGION WIDTH) of (WINDOWPROP mainWindow 'REGION))
                                      (PLUS 10 (HEIGHTIFWINDOW (TIMES 2 YSIZE)
                                                      T)))
                             "Eqn edit" NIL T)))
         (ATTACHWINDOW editWindow mainWindow 'BOTTOM 'LEFT 'LOCALCLOSE)

          (* note%: edit window is justified -- later may automatically reshape it as new text is entered)

         (WINDOWADDPROP editWindow 'CLOSEFN 'DETACHWINDOW)        (* edit window will be detached when closed)

          (* * allow main and edit windows to be independently reshaped)

         (WINDOWDELPROP editWindow 'PASSTOMAINCOMS 'SHAPEW)
         (WINDOWADDPROP editWindow 'REJECTMAINCOMS 'SHAPEW)

          (* * make windows snuggle after reshape)

         (WINDOWADDPROP editWindow 'RESHAPEFN 'EQN.SnuggleWindows)
         (WINDOWADDPROP mainWindow 'RESHAPEFN 'EQN.SnuggleWindows)
         editWindow])

(EQN.SetUpEdit
  [LAMBDA (editWindow mainWindow eqnObj continueFlg piece#)   (* thh%: " 5-Dec-85 10:59")
                                                            (* sets up props for sub edit)
    (EQN.ResultWindow editWindow mainWindow)
    (EQN.ResultObj editWindow eqnObj)
    (EQN.ContinueFlg editWindow continueFlg)
    (EQN.PieceNumber editWindow piece#)
    [WINDOWPROP editWindow 'TEDIT.MENU.COMMANDS '(Find Looks Substitute Character% Looks (Equation
                                                                                             'EQN.Equation)
                                                          (Exit 'Quit "exit from equation editor"
                                                                (SUBITEMS (Next% Piece 'EQN.NextPiece)
                                                                       (Finish% Eqn 'EQN.FinishEqn)
                                                                       (Abort 'EQN.NoUpdateAbort "Terminates eqn
                                                                              editor without changing eqn."]
    (EQN.ObjEditWindow eqnObj editWindow])

(EQN.StartEdit
  [LAMBDA (eqnObj mainWindow piece# continueFlg initialSEL)   (* thh%: " 6-Dec-85 11:21")

          (* Starts edit of specified piece of eqnObj which is currently in TEdit mainWindow.
          Starts new edit only if piece exists.)
                                                            (* continueFlg determines action on normal exit)
                                                            (* initialSEL is initial selection or char number)
    (EQN.AbortEdit mainWindow)                              (* abort any previous eqn edit)
    (COND
       ((AND (IGREATERP piece# 0)
             (ILEQ piece# (EQIO.NumPieces eqnObj)))
        (PROG ((data (EQIO.EqnData eqnObj piece#))
               editWindow editStream box len)
              (SETQ box (FS.Box data (DECODE/WINDOW/OR/DISPLAYSTREAM mainWindow)))
              (SETQ editWindow (EQN.MakeEditWindow mainWindow (fetch XSIZE of box)
                                      (fetch YSIZE of box)))
              [TEDIT NIL editWindow NIL (APPEND '(QUITFN EQN.FinishEdit AFTERQUITFN EQN.ContinueEdit CHARFN
                                                          EQN.CharFn)
                                               (LIST 'PROMPTWINDOW (OR (TEXTPROP (TEXTSTREAM mainWindow)
                                                                              'PROMPTWINDOW)
                                                                        (GETPROMPTWINDOW mainWindow]
                                                            (* sub edit uses same prompt window as main editor)
              (EQN.SetUpEdit editWindow mainWindow eqnObj continueFlg piece#)
              (SETQ editStream (TEXTSTREAM editWindow))
              (SETQ len (FS.Insert data editStream))
              (COND
                 ((EQ initialSEL 'PENDINGDEL)
                  (TEDIT.SETSEL editStream 1 len 'RIGHT T))
                 ((FIXP initialSEL)
                  (TEDIT.SETSEL editStream initialSEL 0 'RIGHT))
```

```
                ((type? SELECTION initialSEL)
                 (TEDIT.SETSEL editStream initialSEL)))
             (EQN.EditWindow mainWindow editWindow)
             [COND
                ((NOT (EQN.SubEditorP mainWindow))

         (* this is a main edit window -- make sure sub edits are aborted when main edit is done
         (don't call EQN.FinishEdit since want QUITFN to return NIL to allow user to confirm quit at top level))

                 (TEXTPROP (TEXTSTREAM mainWindow)
                        'QUITFN
                        'EQN.AbortEdit]                          (* old TEdit version%: (WINDOWPROP mainWindow
                                                                 (QUOTE TEDIT.QUITFN) (QUOTE EQN.FinishEdit)))
         ])
```

## (EQN.StartNextEdit

```
  [LAMBDA (eqnObj mainWindow newPiece# continueFlg initialSEL)   (* thh%: "29-May-85 10:10")
                                                                 (* if another piece of eqnObj exists, starts an edit of it)
    (EQN.StartEdit eqnObj mainWindow newPiece# continueFlg initialSEL)
                                                                 (* mark obj as changed to allow display indication of new piece
                                                                 being edited)
    (TEDIT.OBJECT.CHANGED (TEXTSTREAM mainWindow)
           eqnObj])
```

## (EQN.UpdateEdit

```
  [LAMBDA (mainWindow noChangeFLG)                               (* thh%: " 6-Dec-85 09:52")
                                                                 (* updates sub edit in mainWindow)

         (* noChangeFLG non-NIL means main eqn not changed, but still must notify TEdit obj is changed to update display, e.g.
         uninvert edited piece)

    (PROG ((editWindow (EQN.EditWindow mainWindow))
           value datum eqnObj piece# ptr)
          (COND
             ((WINDOWP editWindow)
              (SETQ eqnObj (EQN.ResultObj editWindow))
              [COND
                 ((NOT noChangeFLG)
                  (EQN.UpdateEdit editWindow)                    (* get updates of any sub edits)
                  (SETQ piece# (EQN.PieceNumber editWindow))
                  (SETQ value (FS.Extract (TEXTSTREAM editWindow)))
                  (SETQ datum (EQIO.EqnDataList eqnObj))
                  (SETQ ptr (NTH datum piece#))
                  (COND
                     (ptr (RPLACA ptr value))
                     (T                                          (* put value on end of datum)
                        (EQIO.SetDataList eqnObj (NCONC1 datum value]
              (EQN.CheckWindowSize mainWindow eqnObj)
              (TEDIT.OBJECT.CHANGED (TEXTSTREAM mainWindow)
                     eqnObj])
```

## (EQN.DefaultData

```
  [LAMBDA (type fontSpec numPieces dataList)                     (* thh%: " 6-Dec-85 09:44")
                                                                 (* gets list of default data items to use for equation specified
                                                                 type)
                                                                 (* currently just a single item -- blank)

         (* if dataList is specified its values are used as the default data --
         either directly if the item is a format string or with default font for the piece if it is a string)

    [COND
       ((NOT numPieces)
        (SETQ numPieces (EQIO.GetInfo type 'numPieces]
    (PROG [(initialData (EQIO.GetInfo type 'initialData]
          (RETURN
           (COND
              ((AND initialData (LITATOM initialData))
               (APPLY* initialData fontSpec type numPieces dataList))
              (T (COND
                    ((NLISTP initialData)
                     (SETQ initialData NIL)))
                 (PROG ((fontNumber (EQN.EquationFontNumber fontSpec)))
                       (RETURN (COND
                                  [numPieces (for i from 1 to numPieces bind initial item
                                                collect (SETQ initial (pop initialData))
                                                      (SETQ item (pop dataList))
                                                                 (* each piece of the equation consists of a single-item format
                                                                 string (unless specified otherwise by value in dataList))
                                                      (COND
                                                         ((AND item (LISTP item))
                                                          item)
                                                         ((IMAGEOBJP item)
                                                          (LIST item))
                                                         (T (LIST (FS.MakeItem [EQN.EquationFont
                                                                                          (PLUS fontNumber (COND
                                                                                                              ((FIXP
                                                                                                                    initial
                                                                                                                    )
                                                                                                               initial)
                                                                                                              (T 0]
                                                                                   (COND
                                                                                      (item (MKSTRING item))
                                                                                      (T " "]
                                          (T NIL])
```

```
(EQN.TypeMenu
  [LAMBDA NIL                                                         (* THH " 1-Jul-85 08:42")
                                                                      (* returns menu of equation types)

    (COND
       ((AND (BOUNDP 'EquationTypeMenu)
             (type? MENU EquationTypeMenu))
        EquationTypeMenu)
       (T                                                             (* compute menu from EquationInfo)
          (SETQ EquationTypeMenu (create MENU
                                         ITEMS _ (SORT (for item in EquationInfo bind label
                                                          collect (SETQ label (EQIO.GetInfo item 'menuLabel))
                                                                (COND
                                                                   (label (LIST label (KWOTE item)))
                                                                   (T item)))
                                                       T)
                                         TITLE _ "Equation Types"])
)
```

;;; hooks to control behavior of equation subeditor

```
(DEFINEQ

(EQN.Equation
  [LAMBDA (textStream)                                                (* thh%: "31-May-85 08:58")
                                                                      (* allows insertion of an equation at current selection in
                                                                      textStream)

    (PROG ((type (MENU (EQN.TypeMenu)))
           currentFont window eqnObj props)
          (COND
             (type (SETQ currentFont (FS.ExtractFont textStream))
                   (SETQ window (EQN.WindowFromText textStream))
                   (SETQ props (EQIO.GetInitialProps type))
                   (SETQ eqnObj (EQIO.Create type (EQN.DefaultData type currentFont (LISTGET props 'numPieces))
                                           currentFont props))
                   (EQN.CheckWindowSize window eqnObj)
                   (TEDIT.INSERT.OBJECT eqnObj textStream)
                   (EQN.StartEdit eqnObj window 1 T 'PENDINGDEL)      (* must mark obj changed to allow display to indicate obj is
                                                                      being edited)

          (* TEDIT.OBJECT.CHANGED can not be called directly from EQN.StartEdit because that fcn is called by the button fcn)

                   (TEDIT.OBJECT.CHANGED textStream eqnObj])

(EQN.NextPiece
  [LAMBDA (textStream)                                                (* thh%: "29-May-85 10:11")
                                                                      (* aborts edit and continues with next piece of eqn)

    (PROG ((editWindow (EQN.WindowFromText textStream))
           mainWindow)
          (SETQ mainWindow (EQN.ResultWindow editWindow))
          (COND
             ((WINDOWP mainWindow)
              (EQN.ContinueFlg editWindow (COND
                                             ((EQN.ContinueFlg editWindow))
                                             (T 1)))
              (EQN.StopEdit mainWindow])

(EQN.FinishEqn
  [LAMBDA (textStream)                                                (* thh%: " 3-May-85 11:53")
                                                                      (* aborts edit without any continuation)

    (PROG ((editWindow (EQN.WindowFromText textStream))
           mainWindow)
          (SETQ mainWindow (EQN.ResultWindow editWindow))
          (COND
             ((WINDOWP mainWindow)
              (EQN.AbortEdit mainWindow])

(EQN.NoUpdateAbort
  [LAMBDA (textStream)                                                (* thh%: " 5-Dec-85 11:21")
                                                                      (* aborts equation editor without updating eqn in main window)
    (EQN.PreventUpdate (EQN.WindowFromText textStream))
    (EQN.FinishEqn textStream])

(EQN.PreventUpdate
  [LAMBDA (window)                                          (* thh%: " 5-Dec-85 11:21")
                                                            (* prevents any update of this or any subeditors)
    (COND
       ((WINDOWP window)
        (WINDOWPROP window 'NOUPDATE T)
        (EQN.PreventUpdate (EQN.EditWindow window])

(EQN.CharFn
  [LAMBDA (textObj charcode)                                (* thh%: "19-Aug-85 08:31")
                                                            (* prevents control chars from being inserted into the document)

          (* allows any char that can be used in a format string and a few special editing chars%: backspace & delete)
                                                            (* allows NEXT syntax char to exit from editor)
    (LET ((syntax (EQN.TEditSpecialChar textObj charcode)))
         (COND
            ((EQ syntax 'NEXT)                              (* next syntax key pressed -- see if this should quit eqn editor)
             (COND
                ((TEDIT.FIND (TEXTSTREAM textObj)
                        ">>*<<" NIL NIL T)                  (* special char -- let TEdit advance to next slot)
                 T)
                (T                                          (* force exit)
                   (EQN.FinishEdit (EQN.WindowFromText textObj))  (* TEDIT.QUIT does not automatically call the QUITFN)
                   (TEDIT.QUIT (TEXTSTREAM textObj))
                   NIL)))
            (syntax                                         (* special character)
                   T)
            ((FS.AllowedChar charcode)                      (* this character can be in format strings)
             T)
            (T                                              (* don't allow this char to be inserted)
               (FLASHWINDOW (EQN.WindowFromText textObj))
               NIL])

(EQN.TEditSpecialChar
  [LAMBDA (textObj charcode)                                (* thh%: "16-Aug-85 09:35")

          (* if charcode is a control character for edit specified by textObj returns its syntax class, else NIL)

          (* this may not correspond exactly to the procedure TEdit uses to determine syntax of a character --
          need to also check terminal table??)

    (LET ((table (OR (READTABLEP (TEXTPROP textObj 'READTABLE))
                     TEDIT.READTABLE))
          syntax)
         (SETQ syntax (TEDIT.GETSYNTAX charcode table))
         (COND
            ((EQ syntax 'NONE)
             (SETQ syntax NIL)))
         syntax])

(EQN.SnuggleWindows
  [LAMBDA (window)                                          (* thh%: " 6-Dec-85 12:07")
                                                            (* reshape fn for independently reshapeable attached windows)

          (* * move attached windows)

    (REPOSITIONATTACHEDWINDOWS window)                      (* Does not work correctly when window itself has attached
                                                            windows ??)

          (* * move main window)

    (EQN.SnuggleMainWindow window])

(EQN.SnuggleMainWindow
  [LAMBDA (window)                                          (* thh%: " 6-Dec-85 11:44")
                                                            (* moves all windows in main window chain)
    (LET ((mainW (MAINWINDOW window))
          region)
         (COND
            ((AND (WINDOWP mainW)
                  (NOT (EQ mainW window)))
             (SETQ region (WINDOWREGION window))            (* position main window above window)

          (* note that MOVEW must be rejected by window so that moving main window wont also move attached window)

             (RESETLST
                 (RESETSAVE (WINDOWADDPROP window 'REJECTMAINCOMS 'MOVEW)
                        `(WINDOWDELPROP %, window REJECTMAINCOMS MOVEW))
                 (MOVEW mainW (fetch (REGION LEFT) of region)
                        (PLUS (fetch (REGION BOTTOM) of region)
                                     (fetch (REGION HEIGHT) of region))))
                 (EQN.SnuggleMainWindow mainW])

)


;;; functions to handle equation fonts

(DEFINEQ

(EQN.EquationFontNumber
  [LAMBDA (fontSpec)                                          (* thh%: "31-Jul-85 08:48")
                                                              (* returns number of the font to use for normal size parts of the
                                                              equation)
                                                              (* fontSpec can be a number which then corresponds to the size
    of the font)
    (COND
       ((NOT fontSpec)
        (SETQ fontSpec DEFAULTFONT)))
    (PROG [(size (OR (NUMBERP fontSpec)
                     (FONTPROP fontSpec 'SIZE]
          (RETURN (for i from 1 to (ARRAYSIZE EquationFontSpecs)
                     smallest (ABS (DIFFERENCE (FONTPROP (ELT EquationFontSpecs i)
                                                         'SIZE)
                                             size])

(EQN.EquationFont
  [LAMBDA (n)                                                 (* thh%: " 5-Dec-85 11:26")
                                                              (* returns equation font number n)
    (COPY (ELT EquationFontSpecs (MAX 1 (MIN (ARRAYSIZE EquationFontSpecs)
                                                 n])

(EQN.GetEqnFont
  [LAMBDA (fontSpec)                                          (* thh%: "31-Jul-85 08:52")
    (EQN.EquationFont (EQN.EquationFontNumber fontSpec])

(EQN.MakeFS
  [LAMBDA (item fontSpec)                                     (* thh%: "31-Jul-85 08:58")
                                                              (* constructs a single element format string)
    (COND
       ((IMAGEOBJP item)
        (LIST item))
       ((LISTP item)
        item)
       (T (LIST (FS.MakeItem fontSpec (MKSTRING item])
)


;;; utilities

(DEFINEQ

(EQN.AdjustWindow
  [LAMBDA (editWindow dWidth dHeight)                         (* thh%: " 6-Dec-85 12:09")
                                                              (* reshapes subeditor window to have extra width and height)
    (PROG ((region (WINDOWPROP editWindow 'REGION))
           (attachedWindows (ATTACHEDWINDOWS editWindow))
           newHeight howAttached)
          (SETQ newHeight (MAX (HEIGHTIFWINDOW 10 T)
                               (PLUS (fetch HEIGHT of region)
                                     dHeight)))

          (* want to shape this window only, not any attached windows --
          depends on correctly setting props when edit windows created)
                                                              (* SETQ howAttached (for window in attachedWindows collect
                                                              (DETACHWINDOW window)))
          (SHAPEW editWindow (create REGION
                                    LEFT _ (fetch LEFT of region)
                                    BOTTOM _ (DIFFERENCE (fetch BOTTOM of region)
                                                    (DIFFERENCE newHeight (fetch HEIGHT of region)))
                                    WIDTH _ (MAX (WIDTHIFWINDOW 10)
                                                 (PLUS (fetch WIDTH of region)
                                                       dWidth))
                                    HEIGHT _ newHeight))      (* reattach the windows -- note that LOCALCLOSE is assumed
                                                              for all windows)
                                                              (* for window in attachedWindows as how in howAttached do
                                                              (ATTACHWINDOW window editWindow
                                                              (CAR how) (CDR how) (QUOTE LOCALCLOSE)))
      ])

(EQN.CheckWindowSize
  [LAMBDA (window eqnObj)                                     (* thh%: "29-May-85 08:52")
```

```
            (* makes sure window can contain new obj -- currently only checks based on height of new obj, assuming YDESC not too
            large)

    (COND
       ((EQN.SubEditorP window)                                        (* only adjust for subeditors (They contain a single line of text))
        (PROG ((box (APPLY* (IMAGEOBJPROP eqnObj 'IMAGEBOXFN)
                            eqnObj
                            (DECODE/WINDOW/OR/DISPLAYSTREAM window)))
               (height (WINDOWPROP window 'HEIGHT))
               extraHeight)
              (SETQ extraHeight (DIFFERENCE (MIN 400 (TIMES 2 (fetch YSIZE of box)))
                                       height))

            (* for now require window to be twice as high as object so don't have to check YDESC --
            should work for most cases)

              (COND
                 ((IGREATERP extraHeight 0)
                  (EQN.AdjustWindow window 0 extraHeight])

)

(DEFINEQ

(EQN.SubEditorP
  [LAMBDA (window)                                                     (* thh%: "28-May-85 09:27")
                                                                       (* non-NIL if window has a eqn subeditor running in it)
    (EQN.ResultWindow window])

(EQN.WindowFromText
  [LAMBDA (textObjORStream)                                            (* thh%: "28-Jun-85 14:32")
                                                                       (* gets window corresponding to a text object or stream)
                                                                       (* note%: \WINDOW field actually is a list whose only element is
    the window)
    (LET [(w (fetch \WINDOW of (TEXTOBJ textObjORStream]
         (OR (WINDOWP w)
             (WINDOWP (CAR w))
             (ERROR "EQN.WindowFromText: unable to find window for textobj/stream = " textObjORStream])

(EQN.EditWindow
  [LAMBDA window                                                       (* thh%: " 3-May-85 08:55")
                                                                       (* returns or sets window of any sub edit)
    (COND
       [(IEQP window 1)
        (PROG [(w (WINDOWPROP (ARG window 1)
                         'EditWindow]                                  (* test for valid window)
              (RETURN (COND
                         ((WINDOWP w)
                          (COND
                             ((AND (OPENWP w)
                                   (EQ (EQN.ResultWindow w)
                                       (ARG window 1)))
                              w)
                             (T (WINDOWPROP (ARG window 1)
                                       'EditWindow NIL)
                                NIL)))
                         (T NIL]
       ((IEQP window 2)
        (WINDOWPROP (ARG window 1)
               'EditWindow
               (ARG window 2)))
       (T NIL])

(EQN.ResultWindow
  [LAMBDA editWindow                                                   (* THH " 2-May-85 16:20")
                                                                       (* returns or sets main window for a sub edit)
    (COND
       ((IEQP editWindow 1)
        (WINDOWPROP (ARG editWindow 1)
               'ResultWindow))
       ((IEQP editWindow 2)
        (WINDOWPROP (ARG editWindow 1)
               'ResultWindow
               (ARG editWindow 2)))
       (T NIL])

(EQN.ResultObj
  [LAMBDA editWindow                                                   (* THH " 2-May-85 16:25")
                                                                       (* returns or sets object being edited)
    (COND
       ((IEQP editWindow 1)
        (WINDOWPROP (ARG editWindow 1)
               'ResultObj))
       ((IEQP editWindow 2)
        (WINDOWPROP (ARG editWindow 1)
               'ResultObj
               (ARG editWindow 2)))
       (T NIL])
```

(**EQN.PieceNumber**

```
  [LAMBDA editWindow                                                  (* THH " 2-May-85 16:37")
                                                                      (* returns or sets number of piece being edited)
    (COND
       ((IEQP editWindow 1)
        (WINDOWPROP (ARG editWindow 1)
               'PieceNumber))
       ((IEQP editWindow 2)
        (WINDOWPROP (ARG editWindow 1)
               'PieceNumber
               (ARG editWindow 2)))
       (T NIL])
```

(**EQN.ContinueFlg**

```
  [LAMBDA editWindow                                                  (* THH " 2-May-85 16:25")
                                                                      (* returns or sets continuation flag)
                                                                      (* THH " 2-May-85 13:35")
    (COND
       ((IEQP editWindow 1)
        (WINDOWPROP (ARG editWindow 1)
               'ContinueFlg))
       ((IEQP editWindow 2)
        (WINDOWPROP (ARG editWindow 1)
               'ContinueFlg
               (ARG editWindow 2)))
       (T NIL])
```

(**EQN.ValidEditWindow**

```
  [LAMBDA (editWindow eqnObj)                                         (* THH " 8-May-85 09:44")

           (* returns editWindow if it is a window currently being used to edit eqnObj, else NIL)

    (COND
       ((AND (WINDOWP editWindow)
             (OPENWP editWindow)
             (EQ (EQN.ResultObj editWindow)
                 eqnObj))
        editWindow)
       (T NIL])
```

(**EQN.ObjEditWindow**

```
  [LAMBDA eqn                                                         (* THH " 8-May-85 10:01")
                                                                      (* gets or sets edit window for eqnObj)
                                                                      (* eqn is of the form (eqnObj {newEditWindow}))
    (PROG (editWindow eqnObj)
          (COND
             ((AND (IGREATERP eqn 0)
                   (SETQ eqnObj (ARG eqn 1))
                   (IMAGEOBJP eqnObj))
              (RETURN (COND
                         ((IEQP eqn 1)
                          (SETQ editWindow (IMAGEOBJPROP eqnObj 'editWindow))
                          (COND
                             ((EQN.ValidEditWindow editWindow eqnObj))
                             (editWindow                              (* remove invalid edit window prop)
                                    (IMAGEOBJPROP eqnObj 'editWindow NIL)
                                    NIL)
                             (T NIL)))
                         ((IEQP eqn 2)
                          (SETQ editWindow (EQN.ValidEditWindow (ARG eqn 2)
                                                  eqnObj))
                          (IMAGEOBJPROP eqnObj 'editWindow editWindow])
)

(DEFINEQ
```

(**EQN.Make**

```
  [LAMBDA (type dataList fontSpec PROPS)                              (* thh%: " 9-Jan-86 10:12")

           (* creates equation of specified type with given dataList -- for variable piece eqns PROPS should include numPieces)

    (COND
       ((EQIO.IsDefined type)
        (LET [(numPieces (LISTGET PROPS 'numPieces]
             (EQIO.Create type (EQN.DefaultData type fontSpec (COND
                                                                       ((EQIO.GetInfo type 'variable?)
                                                                        numPieces)
                                                                       (numPieces (ERROR "Can't specify numPieces for
                                                                                         type = " type))
                                                                       (T NIL))
                                             dataList)
                         fontSpec PROPS)))
       (T (ERROR "Unknown equation type = " type])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS EquationFontSpecs)
)

(RPAQ EquationFontSpecs (READARRAY-FROM-LIST 3 'POINTER 1 '((TimesRoman 8)
                                                              (TimesRoman 10)
                                                              (TimesRoman 12)
                                                              NIL)))

(PUTPROPS EQN.ObjEditWindow ARGNAMES (NIL (eqnObj {newEditWindow})
                                          args))

(PUTPROPS EQN.ContinueFlg ARGNAMES (NIL (editWindow {continueFlg})
                                        args))

(PUTPROPS EQN.PieceNumber ARGNAMES (NIL (editWindow {pieceNumber})
                                        args))

(PUTPROPS EQN.ResultObj ARGNAMES (NIL (editWindow {resultObj})
                                      args))

(PUTPROPS EQN.ResultWindow ARGNAMES (NIL (editWindow {resultWindow})
                                         args))

(PUTPROPS EQN.EditWindow ARGNAMES (NIL (window {editWindow})
                                       args))

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA EQN.ObjEditWindow EQN.ContinueFlg EQN.PieceNumber EQN.ResultObj EQN.ResultWindow EQN.EditWindow)
)
```

;;; FORMATSTRING module: Part 3 of 3

```
(DEFINEQ

(FS.Box
  [LAMBDA (data imageStream)                                          (* thh%: "27-May-87 10:02")

               (* determines box within which data will be displayed on imageStream --
               data is a list whose elements are either image objs or lists of the form
               (FontSpec String Shift))
                                                                      (* added adjustment for shift -- assumes data still spans
                                                                      baseline)

    (for item in data bind (width _ 0)
                           (ascent _ 0)
                           (descent _ 0)
                           (scale _ (DSPSCALE NIL imageStream))
       do (PROG ((itemWidth 0)
                 (itemAscent 0)
                 (itemDescent 0)
                 fullFont imageBox shift)
                [COND
                   ((IMAGEOBJP item)
                    (SETQ imageBox (APPLY* (IMAGEOBJPROP item 'IMAGEBOXFN)
                                          item imageStream))
                    (SETQ itemWidth (fetch XSIZE of imageBox))
                    (SETQ itemAscent (DIFFERENCE (fetch YSIZE of imageBox)
                                            (fetch YDESC of imageBox)))
                    (SETQ itemDescent (fetch YDESC of imageBox)))
                   ((FS.RealStringP item)
                    (SETQ fullFont (FONTCREATE (FS.ItemFont item)
                                          NIL NIL NIL imageStream))
                    (SETQ shift (TIMES scale (FS.ItemShift item)))
                    (SETQ itemWidth (STRINGWIDTH (FS.ItemValue item)
                                           fullFont))
                    [SETQ itemAscent (PLUS shift (FONTPROP fullFont 'ASCENT]
                    (SETQ itemDescent (DIFFERENCE (FONTPROP fullFont 'DESCENT)
                                             shift]
                (add width itemWidth)
                (SETQ ascent (MAX ascent itemAscent))
                (SETQ descent (MAX descent itemDescent)))
       finally (RETURN (create IMAGEBOX
                          XSIZE _ width
                          YSIZE _ (PLUS ascent descent)
                          YDESC _ descent
                          XKERN _ 0])
```

(**FS.Copy**
```
  [LAMBDA (data)                                              (* THH "21-May-85 12:38")

            (* constructs a copy of data -- data is a list whose elements are either image objs or lists of the form
            (FontSpec String))

    (for item in data collect (COND
                                 ((IMAGEOBJP item)            (* note%: COPYALL doesn't call imageobj copyfn so it can't be
                                                              used here)
                                  (APPLY* (IMAGEOBJPROP item 'COPYFN)
                                         item))
                                 (T (COPYALL item])
```

(**FS.Display**
```
  [LAMBDA (data imageStream invert?)                          (* thh%: "27-May-87 10:02")

            (* displays data on imageStream -- data is a list whose elements are either image objs or lists of the form
            (FontSpec String Shift))
                                                              (* added shift)
    (PROG ((scale (DSPSCALE NIL imageStream))
           xStart yStart)
          [COND
             (invert? (SETQ xStart (DSPXPOSITION NIL imageStream))
                    (SETQ yStart (DSPYPOSITION NIL imageStream]
          [for item in data do (COND
                                  ((IMAGEOBJP item)
                                   (PROG ((xPos (DSPXPOSITION NIL imageStream))
                                          (yPos (DSPYPOSITION NIL imageStream)))
                                         (APPLY* (IMAGEOBJPROP item 'DISPLAYFN)
                                                item imageStream)
                                         (MOVETO (PLUS xPos (fetch XSIZE of (APPLY* (IMAGEOBJPROP item 'IMAGEBOXFN)
                                                                                     item imageStream)))
                                                yPos imageStream)))
                                  ((FS.RealStringP item)
                                   (PROG [(oldFont (DSPFONT NIL imageStream))
                                          (shift (TIMES scale (FS.ItemShift item]
                                         (DSPFONT (FS.ItemFont item)
                                                imageStream)
                                         (COND
                                            ((NOT (ZEROP shift))
                                             (RELMOVETO 0 shift imageStream)))
                                         (PRIN1 (FS.ItemValue item)
                                                imageStream)
                                         (DSPFONT oldFont imageStream)
                                         (COND
                                            ((NOT (ZEROP shift))
                                             (RELMOVETO 0 (MINUS shift)
                                                    imageStream]
          (COND
             (invert? (PROG ((box (FS.Box data imageStream)))
                            (DSPFILL (create REGION
                                            LEFT _ xStart
                                            BOTTOM _ (DIFFERENCE yStart (fetch YDESC of box))
                                            WIDTH _ (fetch XSIZE of box)
                                            HEIGHT _ (fetch YSIZE of box))
                                   BLACKSHADE
                                   'INVERT imageStream])
```

(**FS.Get**
```
  [LAMBDA (fileStream)                                        (* thh%: "13-Jun-85 09:17")

            (* reads data from fileStream -- data is a list whose elements are either image objs or lists of the form
            (FontSpec String))
                                                              (* also reads a list of such data items)
    (HREAD fileStream])
```

(**FS.Put**
```
  [LAMBDA (data fileStream)                                   (* THH " 2-May-85 14:31")

            (* puts data on fileStream -- data is a list whose elements are either image objs or lists of the form
            (FontSpec String))
                                                              (* also writes a list of such data items)
    (HPRINT data fileStream])
```

```
(FS.ItemFont
  [LAMBDA (dataItem)                                           (* thh%: "15-May-85 11:29")
                                                               (* returns font spec of single data item)

    (COND
       ((LISTP dataItem)
        (CAR dataItem))
       (T NIL])


(FS.ItemValue
  [LAMBDA (dataItem)                                           (* THH "21-May-85 12:28")
                                                               (* returns string of single data item or image object)

    (COND
       ((LISTP dataItem)
        (CADR dataItem))
       ((IMAGEOBJP dataItem)
        dataItem])


(FS.ItemShift
  [LAMBDA (dataItem)                                           (* thh%: "27-May-87 09:57")
                                                               (* shift is number of points to move up)

    (COND
       ((LISTP dataItem)
        (OR (CADDR dataItem)
            0))
       (T 0])


(FS.MakeItem
  [LAMBDA (fontSpec string shift)                              (* thh%: "27-May-87 09:48")
                                                               (* makes data item from fontSpec and string with shift)

    (COND
       ((AND shift (NOT (ZEROP shift)))
        (LIST fontSpec string shift))
       (T (LIST fontSpec string])


(FS.Extract
  [LAMBDA (stream)                                             (* thh%: "27-May-87 10:23")
                                                               (* extracts data from TEdit stream)
    (SETFILEPTR stream 0)                                      (* BIN gets imageObj or individual characters)
    (PROG ((EOFptr (GETEOFPTR stream))
           (CHcount 0)
           data result)
          [SETQ data (while (ILESSP CHcount EOFptr) collect (add CHcount 1)
                                                          (CONS CHcount (BIN stream]
                                                               (* data has (ch# . item))
                                                               (* now combine characters into strings and get font information)
          [SETQ data (for item in data collect item unless (AND (NUMBERP (CDR item))
                                                                 (NOT (FS.AllowedChar (CDR item]
                                                               (* remove control characters, e.g.
                                                               CR, LF, tab...)
          [SETQ result (while data bind item fontSpec shift
                          collect (SETQ item (CAR data))
                                  (SETQ data (CDR data))

          (* note that imageobjs are not copied so changes to returned obj will also change original)

                                  (COND
                                     ((IMAGEOBJP (CDR item)))
                                     (T (SETQ fontSpec (FS.ExtractFont stream (CAR item)))
                                        (SETQ shift (FS.ExtractShift stream (CAR item)))
                                        (PROG ([string (MKSTRING (CHARACTER (CDR item]
                                               (nextItem (CAR data)))
                                              (while [AND nextItem (NOT (IMAGEOBJP (CDR nextItem)))
                                                          (EQUAL fontSpec (FS.ExtractFont stream (CAR nextItem)))
                                                          (EQUAL shift (FS.ExtractShift stream (CAR nextItem]
                                                 do [SETQ string (CONCAT string (CHARACTER (CDR nextItem]
                                                    (SETQ data (CDR data))
                                                    (SETQ nextItem (CAR data)))
                                              (RETURN (FS.MakeItem fontSpec string shift]
          (RETURN (COND
                     ((AND result (FS.RealStringP (CAR result)))   (* first item is a string)
                      result)
                     (T                                        (* preserve initial font by including a null string item at beginning
                        of list)                               (* this item has zero shift)
                        (CONS (FS.MakeItem (FS.ExtractFont stream (COND
                                                                     ((ZEROP EOFptr)
                                                                      NIL)
                                                                     (T 1)))
                                           "")
                              result])


(FS.ExtractFont
  [LAMBDA (stream selOrCh#)                                    (* thh%: "20-Feb-86 14:05")
```

```
                                                                  (* gets font spec for specified selection)
    (PROG ((looks (TEDIT.GET.LOOKS stream selOrCh#)))
          (RETURN (LIST (LISTGET looks 'FAMILY)
                        (LISTGET looks 'SIZE)
                        (LIST (LISTGET looks 'WEIGHT)
                              (LISTGET looks 'SLOPE)
                              (LISTGET looks 'EXPANSION])
```

(**FS.ExtractShift**

```
  [LAMBDA (stream selOrCh#)                                       (* thh%: "27-May-87 10:26")
                                                                  (* gets upward shift in points for specified selection)
    (LET ((looks (TEDIT.GET.LOOKS stream selOrCh#))
          shift)
         (COND
            ((SETQ shift (LISTGET looks 'SUPERSCRIPT))
             shift)
            ((SETQ shift (LISTGET looks 'SUBSCRIPT))
             (MINUS shift))
            (T 0])
```

(**FS.Insert**

```
  [LAMBDA (data stream)                                           (* thh%: "27-May-87 10:43")
                                                                  (* inserts data list into TEdit stream)
    (for item in data bind (length _ 0)
                           shift
       do [COND
             ((IMAGEOBJP item)
              (add length 1)
              (TEDIT.INSERT.OBJECT item stream))
             [(FS.RealStringP item)
              (add length (NCHARS (FS.ItemValue item)))
              (SETQ shift (FS.ItemShift item))
              (COND
                 [(ZEROP shift)
                  (TEDIT.INSERT stream (FS.ItemValue item)
                         NIL
                         (FONTCREATE (FS.ItemFont item]
                 [(MINUSP shift)
                  (TEDIT.INSERT stream (FS.ItemValue item)
                         NIL
                         `(FONT ,(FS.ItemFont item)
                                SUBSCRIPT
                                ,(MINUS shift]
                 (T (TEDIT.INSERT stream (FS.ItemValue item)
                           NIL
                           `(FONT ,(FS.ItemFont item)
                                  SUPERSCRIPT
                                  ,shift]
             (T                                                   (* null string -- preserve font info)
                (TEDIT.CARETLOOKS stream (FONTCREATE (FS.ItemFont item]
       finally (RETURN length])
```

(**FS.AllowedChar**

```
  [LAMBDA (charcode)                                              (* thh%: "19-Aug-85 08:29")
                                                                  (* returns T if charcode can be included in format strings)
    (IGEQ charcode (CHARCODE " "])
```

(**FS.RealStringP**

```
  [LAMBDA (item nullOk)                                           (* thh%: "31-Jul-85 08:11")
    (AND (LISTP item)
         (OR nullOk (NOT (EQUAL "" (FS.ItemValue item])
)
```

;;; Now load EQUATIONFORMS

```
(FILESLOAD EQUATIONFORMS)

(PUTPROPS EQUATIONS COPYRIGHT ("Xerox Corporation" 1986 1987 1988))
```

## FUNCTION INDEX

EQIO.AddType ....7
EQIO.AllProps ....6
EQIO.ButtonEventIn ....3
EQIO.Copy ....3
EQIO.CopyList ....3
EQIO.Create ....2
EQIO.CreateFns ....2
EQIO.DefaultSelectFn ....4
EQIO.Display ....2
EQIO.EqnData ....5
EQIO.EqnDataList ....5
EQIO.EqnProperty ....5
EQIO.EqnType ....5
EQIO.Get ....4
EQIO.GetBox ....7
EQIO.GetDataPosition ....8
EQIO.GetDataSelectRegion ....8
EQIO.GetDataSpec ....8
EQIO.GetDataSpecList ....8
EQIO.GetInfo ....7
EQIO.GetInitialProps ....6
EQIO.Imagebox ....2
EQIO.IsDefined ....7
EQIO.MakeDataSpec ....8
EQIO.MakeSelectionMenu ....5
EQIO.MakeSpec ....8
EQIO.NewStructure ....6
EQIO.NumPieces ....6
EQIO.Put ....4
EQIO.ResetTypeProps ....7
EQIO.Selection ....4
EQIO.SelectRegion ....4
EQIO.SetDataList ....5
EQIO.SetInfo ....7
EQIO.Specify ....6
EQIO.TypeProp ....7
EQIO.WhenDeleted ....4
EQN.AbortEdit ....9
EQN.AdjustWindow ....14
EQN.CharFn ....13
EQN.CheckWindowSize ....14
EQN.ContinueEdit ....9
EQN.ContinueFlg ....16
EQN.DefaultData ....11
EQN.EditWindow ....15
EQN.Equation ....12
EQN.EquationFont ....14
EQN.EquationFontNumber ....14
EQN.FinishEdit ....10
EQN.FinishEqn ....12
EQN.GetEqnFont ....14
EQN.Make ....16
EQN.MakeEditWindow ....10
EQN.MakeFS ....14
EQN.NextPiece ....12
EQN.NoUpdateAbort ....12
EQN.ObjEditWindow ....16
EQN.PieceNumber ....16
EQN.PreventUpdate ....13
EQN.ResultObj ....15
EQN.ResultWindow ....15
EQN.SetUpEdit ....10
EQN.SnuggleMainWindow ....13
EQN.SnuggleWindows ....13
EQN.StartEdit ....10
EQN.StartNextEdit ....11
EQN.StopEdit ....9
EQN.SubEditorP ....15
EQN.TEditSpecialChar ....13
EQN.TypeMenu ....12
EQN.UpdateEdit ....11
EQN.ValidEditWindow ....16
EQN.WindowFromText ....15
FS.AllowedChar ....20
FS.Box ....17
FS.Copy ....18
FS.Display ....18
FS.Extract ....19
FS.ExtractFont ....19
FS.ExtractShift ....20
FS.Get ....18
FS.Insert ....20
FS.ItemFont ....19
FS.ItemShift ....19
FS.ItemValue ....19
FS.MakeItem ....19
FS.Put ....18
FS.RealStringP ....20

## PROPERTY INDEX

EQIO.AllProps .....8
EQIO.EqnProperty ..8
EQIO.NumPieces ....8
EQIO.TypeProp .....8
EQN.ContinueFlg ..17
EQN.EditWindow ...17
EQN.ObjEditWindow 17
EQN.PieceNumber ..17
EQN.ResultObj ....17
EQN.ResultWindow .17

## VARIABLE INDEX

EquationDefaultSelectFn ....8
EquationFontSpecs ....17
EquationImageFns ....8
EquationInfo ....8
UnknownEquationData ....8