File created: 12-Apr-2024 19:58:59 **{WMEDLEY}<doctools>IMTEDIT.;5**

edit by: rmk

changes to: (FNS MAKE.IM.DOCUMENT)

previous date: 6-Mar-2024 21:18:02 {WMEDLEY}<doctools>IMTEDIT.;4

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ IMTEDITCOMS
       ((FNS IM.TEDIT DUMP DUMP.HEADERS.FOOTERS DUMP.HRULE CHANGE.FONT IM.BOUT.IMAGEOBJ IM.TEDIT.DUMP.COMMANDS
             IM.TEDIT.DUMP.FOOTNOTES IM.TEDIT.DUMP.PARA INDEXX.PARSE.TYPE FORMAT.DEF FORMAT.LISPWORD
             MAKE.IM.DOCUMENT PRINT.NOTE SEND.INFO)
        (COMS                                                   ; fns for drawing vrules to the left of definition text -- an unused,
                                                                ; never-fully debuged feature
              (FNS IM.VRULE.DISPLAYFN CREATE.VRULE.OBJECT PRINT.VRULES.ON.PAGE)
              (VARS (IM.VRULE.STATE.LIST))
              (INITVARS (IM.VRULE.OBJECT.IMAGEFNS NIL)
                     (IM.PRINT.VRULE.FLG NIL)))
        (COMS                                                   ; fns for printing page numbers
              (FNS IM.FOLIO.DISPLAYFN IM.FOLIO.SIZEFN CREATE.FOLIO.OBJECT GET.FOLIO.STRING)
              (INITVARS (IM.FOLIO.OBJECT.IMAGEFNS NIL)))
        (COMS                                                   ; TOPROG functions, used to define the translating actions of IM
                                                                ; text objects.
              (FNS ARG#TOPROG BIGLISPCODE#TOPROG BRACKET#TOPROG CHAPTER#TOPROG COMMENT#TOPROG DEF#TOPROG
                   FIGURE#TOPROG FN#TOPROG FNDEF#TOPROG FOOT#TOPROG INCLUDE#TOPROG INDEX#TOPROG INDEXX#TOPROG
                   IT#TOPROG LBRACKET#TOPROG LISP#TOPROG LISPCODE#TOPROG LISPWORD#TOPROG LIST#TOPROG
                   MACDEF#TOPROG NOTE#TOPROG PRINT.SPECIAL.CHARS#TOPROG PROPDEF#TOPROG RBRACKET#TOPROG
                   REF#TOPROG RM#TOPROG SUB#TOPROG SUBSEC#TOPROG SUPER#TOPROG TABLE#TOPROG TAG#TOPROG
                   TERM#TOPROG VAR#TOPROG VARDEF#TOPROG)
              (VARS TO.NAME.LIST TO.SYNONYM.LIST)
              (IFPROP (TO.PROG TO.ARGS TO.ARG.SYNONYMS TO.TYPE TO.ARG.TYPE)
                     * TO.NAME.LIST))
        [INITVARS (IM.TEDIT.FONT.DEFS '(NIL (FAMILY MODERN FACE MRR SIZE 10)
                                            FOOTNOTE
                                            (FAMILY MODERN FACE MRR SIZE 10)
                                            NOTE
                                            (FAMILY MODERN FACE MIR SIZE 8)
                                            BOLD
                                            (FAMILY MODERN FACE BRR SIZE 10)
                                            ITALIC
                                            (FAMILY MODERN FACE MIR SIZE 10)
                                            LISP
                                            (FAMILY MODERN FACE BRR SIZE 10)
                                            ARG
                                            (FAMILY MODERN FACE MIR SIZE 10]
        [INITVARS (IM.CHAPTER.TITLE.FONT '(FAMILY MODERN FACE BRR SIZE 18))
               (IM.SUBSEC.ONE.TITLE.FONT '(FAMILY MODERN SIZE 14 FACE BRR))
               (IM.SUBSEC.TWO.TITLE.FONT '(FAMILY MODERN SIZE 12 FACE BRR))
               (IM.SUBSEC.THREE.TITLE.FONT '(FAMILY MODERN SIZE 10 FACE BRR))
               (IM.TEXT.FONT '(FAMILY MODERN FACE MRR SIZE 10))
               (IM.HEADER.FOOTER.FONT '(FAMILY MODERN FACE MRR SIZE 8))
               (IM.XEROX.LOGO.FONT '(FAMILY MODERN FACE BRR SIZE 30]
        (COMS
              ;; the following variables specify all of the lengths used for possitioning IM text, headers, etc.  on the page.  All of these are measured
              ;; with respect to the page *margins* <the region on the page defined by the Tedit margin parameters> or with respect to the page
              ;; *edges* <the edges of the physical page>.

              ;; Note: The formatting and printing does not always position the image on the page exactly as specified.  It will probably be
              ;; necessary to adjust any variables based on the page edges until they come out correctly on your printer.

              ;; indentation of 1st line of definitian header, measured in points from left page margin.  Also used for indentation of hrule under defn
              ;; header.

              (INITVARS (IM.DEF.TITLE.1STLEFTMARGIN 75))

              ;; indentation of 2nd and other overflow lines of definition header, measured in points from left page margin.

              (INITVARS (IM.DEF.TITLE.LEFTMARGIN 204))

              ;; indentation of vertical rule to the left of definition text, measured in points from left page margin.  This is a never-used,
              ;; never-debugged feature.

              (INITVARS (IM.VRULE.X 194))
                                                                ; y-pos of top-left corner of top text line, measured in points from
                                                                ; bottom page edge.
              (INITVARS (IM.TEXT.TOPMARGIN 738))
                                                                ; y-pos of bottom-left corner of bottom text line, measured in
                                                                ; points from bottom page edge.
              (INITVARS (IM.TEXT.BOTTOMMARGIN 54))
                                                                ; x-pos of left edge of text, measured in points from the left page
                                                                ; margin.
              (INITVARS (IM.TEXT.LEFTMARGIN 204))
                                                                ; x-pos of right edge of text, measured in points from the left
                                                                ; page margin.
              (INITVARS (IM.TEXT.RIGHTMARGIN 504))
```

```
        ;; X-pos and Y-pos of the lower-left corner of the '[This page intentionally left blank]' message printed on blank pages, measured in
        ;; points from the left and bottom page edges.

        (INITVARS (IM.BLANKPAGE.SPECIALX 258)
               (IM.BLANKPAGE.SPECIALY 400))

        ;; In the table of contents, indentation of first and second-level subsection headers, measured in points from the left page margin.

        (INITVARS (IM.TOC.SUBSEC.ONE.LEFTMARGIN 120)
               (IM.TOC.SUBSEC.TWO.LEFTMARGIN 216))

        ;; in the index, the indentation of the first line and remaining lines of a top-level entry, of a subentry, and of a subsubentry, measured
        ;; in points from the left page margin <for the left column>.

        (INITVARS (IM.INDEX.1STLEFTMARGIN 0)
               (IM.INDEX.LEFTMARGIN 75)
               (IM.INDEX.SUB.1STLEFTMARGIN 25)
               (IM.INDEX.SUB.LEFTMARGIN 75)
               (IM.INDEX.SUBSUB.1STLEFTMARGIN 50)
               (IM.INDEX.SUBSUB.LEFTMARGIN 75))

        ;; on the title page, the y-pos of the lower-left corner of the first line in the title <and of the XEROX logo>, measured in points from the
        ;; bottom page margin.  The X-pos is always 0 for the XEROX logo, and the normal text indentation for the title.

        (INITVARS (IM.TITLEPAGE.TITLE.Y 258))

        ;; on the title page, the y-pos of the lower-left corner of the first line in the document number, measured in points from the bottom
        ;; page margin.  The Y-pos is always the normal text indentation.

        (INITVARS (IM.TITLEPAGE.DOCNUMBER.Y 45))

        ;; Tedit tab setting used for subsection heading text.  '(40 . LEFT)' determines the indentation of the title after the subsec number,
        ;; measured in points from the left page margin.  '18' is the tab used if the subsec number is wider than 40 pts.

        [INITVARS (IM.SUBSEC.TITLE.TABS '(18 (40 . LEFT]

        ;; Tedit tab setting used for chapter titles, headers, and footers to right-justify text.  '(504 . RIGHT)' specifies a right tab at the
        ;; right-hand edge of the text, measured in points from the left page margin.

        [INITVARS (IM.RIGHT.MARGIN.TABS '(0 (504 . RIGHT]

        ;; Tedit tab setting used for labeled lists, numbered lists, bullet-ed lists.  '(186 . RIGHT)' right-justifies the label on the left of the center
        ;; space.  '(204 . LEFT)' starts the first line of the list item with the same indentation as normal text.  Both measurements are
        ;; measured in points from the left page margin.

        [INITVARS (IM.LABELED.LIST.TABS '(18 (186 . RIGHT)
                                             (204 . LEFT]

        ;; left, right, top, and bottom margins of the 'page region' , measured in points from the four edges of the page.

        (INITVARS (IM.PAGE.LEFTMARGIN 58)
               (IM.PAGE.RIGHTMARGIN 54)
               (IM.PAGE.TOPMARGIN 54)
               (IM.PAGE.BOTTOMMARGIN 54))

        ;; top margin of the page region for the first page of a chapter <where the first paragraph is the chapter title>, measured in points from
        ;; the top page edge.

        (INITVARS (IM.PAGE.FIRST.TOPMARGIN 12))

        ;; top margin of the page region for the first page of the index, measured in points from the top page edge.  Note that in the case of
        ;; the index, because it uses two columns, the index title is implemented as a Tedit header, instead of as the first paragraph of the
        ;; document.

        (INITVARS (IM.INDEX.PAGE.FIRST.TOPMARGIN 144))
                                                          ; y-pos of lower-left corner of footer text, measured in points from
                                                          ; the bottom page edge.
        (INITVARS (IM.FOOTER.Y 22))
                                                          ; y-pos of the footer hrule, measured in points from the bottom
                                                          ; page edge.
        (INITVARS (IM.FOOTER.RULE.Y 30))
                                                          ; y-pos of lower-left corner of header text, measured in points
                                                          ; from the bottom page edge.
        (INITVARS (IM.HEADER.Y 761))
                                                          ; y-pos of the header hrule, measured in points from the bottom
                                                          ; page edge.
        (INITVARS (IM.HEADER.RULE.Y 757))
                                                          ; y-pos of lower-left corner of bottom draft message, measured in
                                                          ; points from the bottom page edge.
        (INITVARS (IM.DRAFT.MESSAGE.BOTTOM.Y 5))
                                                          ; y-pos of lower-left corner of top draft message, measured in
                                                          ; points from the bottom page edge.
        (INITVARS (IM.DRAFT.MESSAGE.TOP.Y 775))

        ;; x-pos of lower-left corner of both top and bottom draft messages, measured in points from the left page edge.

        (INITVARS (IM.DRAFT.MESSAGE.X 200)))
    (FILES TEDIT IMTRAN HRULE IMINDEX)
    (FNS TRANSLATE.DUMPOUT TRANSLATE.SAVE.DUMPOUT)
    (MACROS IM.HOLD.FOOTNOTES DUMPOUT SAVE.DUMPOUT)))

(DEFINEQ

(IM.TEDIT
  [LAMBDA (INFILE.NAME OUTFILE.FLG)                                    (* mjs " 4-Aug-86 10:53")

        (* * This function takes an IM format file, and produces a formatted Tedit text stream.
        Note that the Tedit text stream is a totally different document ---
        the user may edit it to clear up formatting problems before printing, but the user must be careful not to edit this document
```

without going back and changing the original IM format file. INFILE.NAME is the name of an IM format file.)

(* * If OUTFILE.FLG is NIL, the output file is just sent to the default printer.
If OUTFILE.FLG is T, the outfile textstream is simply returned. If OUTFILE.FLG = anything else, it is taken as a file name to put the press file which is created <but not printed>.)

```
(PROG ((ERRFILE.NAME (PACKFILENAME 'NAME (FILENAMEFIELD INFILE.NAME 'NAME)
                           'EXTENSION
                           'IMERR))
       (PTRFILENAME (PACKFILENAME 'NAME (FILENAMEFIELD INFILE.NAME 'NAME)
                           'EXTENSION
                           'IMPTR))
       ERRFILE DOC.VAL)
      (DECLARE (SPECVARS ERRFILE.NAME ERRFILE))
      (SETQ DOC.VAL (MAKE.IM.DOCUMENT '(IMTRAN INFILE.NAME)
                           OUTFILE.FLG NIL (CONCAT "IMTEDIT Hardcopy of " INFILE.NAME)
                           NIL PTRFILENAME))
      (if (OPENP ERRFILE)
          then (CLOSEF ERRFILE)
               (printout T "Error File: " (FULLNAME ERRFILE)
                     T))
      (RETURN DOC.VAL])
```

(DUMP

```
[LAMBDA (C)                                                    (* mjs "12-Apr-85 10:41")
```

(* * this function dumps the character C into the Tedit stream. It maps multiple CRs into a single CR, and decides when to put out paragraph looks. If C is a list, it is treated as a special "Dump Command" which does things such as changing fonts.)

```
                                                               (* printout T "flg=" IM.TEDIT.CR.FLG ";C="
                                                               (if (AND (LISTP C) (EQ (CAR C) (QUOTE TEXT))) then
                                                               (QUOTE TEXT) elseif (SMALLP C) then
                                                               (CONCAT C "/" (CHARACTER C)) else C) T)
                                                               (* handle all CRs as examples of the Dump Command CR)
   (COND
      [(EQ C (CHARCODE CR))
       (IM.TEDIT.DUMP.COMMANDS '(CR]
      ((ZEROP C)                                               (* flush null chars)
       NIL)
      ((LISTP C)                                               (* treat lists as Dump Commands)
       (IM.TEDIT.DUMP.COMMANDS C))
      [IM.TEDIT.CR.FLG (COND
                          ((EQ C (CHARCODE SPACE))             (* ignore spaces after a CR)
                           (NILL))
                          ((EQ IM.TEDIT.CR.FLG 'ONE)           (* if there was only one CR, put out a space and the following
                                                               char)
                           (BOUT IM.OUTFILE (CHARCODE SPACE))
                           (SETQ IM.TEDIT.CR.FLG NIL)
                           (DUMP C))
                          ((EQ IM.TEDIT.CR.FLG 'MANY)          (* time to start a new para)
                           (SETQ IM.TEDIT.CR.FLG NIL)
                           (DUMP C]
      ((SMALLP C)
       (BOUT IM.OUTFILE C))
      ((IMAGEOBJP C)
       (IM.BOUT.IMAGEOBJ C IM.OUTFILE))
      (T (SHOULDNT])
```

(DUMP.HEADERS.FOOTERS

```
[LAMBDA (HEADER.TEXT FOOTER.TEXT)                              (* mjs "18-Sep-85 15:40")
  [COND
     (HEADER.TEXT (SETQ HEADER.TEXT (U-CASE HEADER.TEXT))
            (DUMPOUT CR CR START.PARA FONT IM.HEADER.FOOTER.FONT PARALOOKS
                   `(TYPE PAGEHEADING SUBTYPE VERSOHEAD QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 RIGHTMARGIN %,
                          IM.TEXT.RIGHTMARGIN)
                   DUMP.CHARS HEADER.TEXT CR CR)
            (DUMP.HRULE 1 NIL `(TYPE PAGEHEADING SUBTYPE VERSOHEADRULE 1STLEFTMARGIN 0 LEFTMARGIN 0
                                     RIGHTMARGIN %, IM.TEXT.RIGHTMARGIN))
            (DUMPOUT CR CR START.PARA FONT IM.HEADER.FOOTER.FONT PARALOOKS
                   `(TYPE PAGEHEADING SUBTYPE RECTOHEAD QUAD RIGHT 1STLEFTMARGIN 0 LEFTMARGIN 0 RIGHTMARGIN %,
                          IM.TEXT.RIGHTMARGIN)
                   DUMP.CHARS HEADER.TEXT CR CR)
            (DUMP.HRULE 1 NIL `(TYPE PAGEHEADING SUBTYPE RECTOHEADRULE 1STLEFTMARGIN 0 LEFTMARGIN 0
                                     RIGHTMARGIN %, IM.TEXT.RIGHTMARGIN]
  (COND
     (FOOTER.TEXT (SETQ FOOTER.TEXT (U-CASE FOOTER.TEXT))
            (DUMP.HRULE 1 NIL `(TYPE PAGEHEADING SUBTYPE VERSOFOOTRULE 1STLEFTMARGIN 0 LEFTMARGIN 0
                                     RIGHTMARGIN %, IM.TEXT.RIGHTMARGIN))
            (DUMPOUT CR CR START.PARA FONT IM.HEADER.FOOTER.FONT PARALOOKS
                   `(TYPE PAGEHEADING SUBTYPE VERSOFOOT QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 TABS %,
                          IM.RIGHT.MARGIN.TABS RIGHTMARGIN %, IM.TEXT.RIGHTMARGIN)
                   DUMP.CHARS
                   (CREATE.FOLIO.OBJECT)
                   TAB DUMP.CHARS FOOTER.TEXT CR CR)
            (DUMP.HRULE 1 NIL `(TYPE PAGEHEADING SUBTYPE RECTOFOOTRULE 1STLEFTMARGIN 0 LEFTMARGIN 0
                                     RIGHTMARGIN %, IM.TEXT.RIGHTMARGIN))
```

```
            (DUMPOUT CR CR START.PARA FONT IM.HEADER.FOOTER.FONT PARALOOKS
                   `(TYPE PAGEHEADING SUBTYPE RECTOFOOT QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 TABS %,
                          IM.RIGHT.MARGIN.TABS RIGHTMARGIN %, IM.TEXT.RIGHTMARGIN)
                   DUMP.CHARS FOOTER.TEXT TAB DUMP.CHARS (CREATE.FOLIO.OBJECT)
                   CR CR])
```

(DUMP.HRULE

```
[LAMBDA (RULE.WIDTH ADDITIONAL.PARA.LEADING SPECIAL.PARALOOKS) (* mjs "18-Sep-85 15:25")
```

(* * old def, used when CR at end of line caused hrule to be too far down%:
(DUMPOUT FONT (QUOTE (FAMILY MODERN FACE MRR SIZE 10)) PARALOOKS SPECIAL.PARALOOKS
PARALOOKS (LIST (QUOTE PARALEADING) (IPLUS -10 (if ADDITIONAL.PARA.LEADING else 0))
(QUOTE LINELEADING) 0) DUMP.CHARS (HRULE.CREATE RULE.WIDTH) CR CR))

```
  (DUMPOUT FONT NIL PARALOOKS SPECIAL.PARALOOKS PARALOOKS (LIST 'PARALEADING (COND
                                                                                (ADDITIONAL.PARA.LEADING)
                                                                                (T 0))
                                                                'LINELEADING 0)
          DUMP.CHARS
          (HRULE.CREATE RULE.WIDTH)
          CR CR])
```

(CHANGE.FONT

```
[LAMBDA (FONT)                                                  (* mjs "11-Apr-85 15:49")
```

(* * changes all of the text between the last font change and the current position to the current font, and changes the current font to FONT. If the current position is the same as that of the last font change <this can happen if you have multiple font changes in a row> just change the current font.)

```
  (PROG ((CURRENT.POS (GETFILEPTR IM.OUTFILE)))
        (COND
           ((NEQ IM.TEDIT.LAST.FONT.BEGIN CURRENT.POS)
            (push IM.CHARLOOKS (COND
                                  ((LISTGET IM.TEDIT.FONT.DEFS IM.TEDIT.FONT))
                                  (T IM.TEDIT.FONT))
                  IM.TEDIT.LAST.FONT.BEGIN
                  (IDIFFERENCE CURRENT.POS IM.TEDIT.LAST.FONT.BEGIN))
                                                    (* be sure to reset Tedit selection after any formatting operation)
            (SETQ IM.TEDIT.LAST.FONT.BEGIN CURRENT.POS)))
        (SETQ IM.TEDIT.FONT FONT])
```

(IM.BOUT.IMAGEOBJ

```
[LAMBDA (OBJ FILE)                                              (* mjs "11-Apr-85 12:09")
  (COND
     ((NOT (IMAGEOBJP OBJ))
      (SHOULDNT))
     (T (PROG [(CURR.CH# (ADD1 (GETFILEPTR FILE]
              (TEDIT.INSERT.OBJECT OBJ FILE CURR.CH#)
              (SETFILEPTR FILE CURR.CH#])
```

(IM.TEDIT.DUMP.COMMANDS

```
[LAMBDA (C)                                                     (* mjs " 1-Oct-85 15:14")
```

(* * this function interpretes Dump Commands to IM.TEDIT.DUMP, which are always lists whose CAR is the command name.)

```
  (SELECTQ (CAR C)
       (TEXT                                                    (* just flush TEX output string)
             NIL)
       (START.PARA
```

(* by setting IM.TEDIT.CR.FLG to NIL, this ensures that any following spaces will not be swollowed because they follow a CR. Warning%: this should only be called after a paragraph is totally ended and finished.)

```
                (COND
                   ((NEQ IM.TEDIT.CR.FLG 'MANY)
                    (IM.ERROR "START.PARA command should only be called after end of paragraph.  Is called
                           when IM.TEDIT.CR.FLG =" IM.TEDIT.CR.FLG)))
                (SETQ IM.TEDIT.CR.FLG NIL))
       (DUMP.FOOTNOTES                                          (* dump out any footnotes without starting new paragraph
                                                                <<which would freeze para formatting info>>)
            (IM.TEDIT.DUMP.FOOTNOTES))
       ((START.SUPER START.SUB)
            (SETQ IM.TEDIT.SUB.SUPER.BEGIN (GETFILEPTR IM.OUTFILE)))
       ((END.SUPER END.SUB)
```

(* this is a very simple scheme --- currently, it does not allow nested super- or subscripts.)

```
           (PROG ((CURRENT.POS (GETFILEPTR IM.OUTFILE)))
                 (push IM.CHARLOOKS (if (EQ (CAR C)
                                            'END.SUPER)
                                        then '(SUPERSCRIPT 3)
```

```
                                    else '(SUBSCRIPT 3))
                            IM.TEDIT.SUB.SUPER.BEGIN
                            (IDIFFERENCE CURRENT.POS IM.TEDIT.SUB.SUPER.BEGIN))))
        (PARALOOKS                                                  (* add para looks to list for next para)
                   (push IM.TEDIT.PARA.LOOKS (CDR C)))
        (CR

          (* if we have recieved at least one CR before, set IM.TEDIT.CR.FLG = MANY, otherwise this is the first CR)

             [COND
                ((EQ IM.TEDIT.CR.FLG 'ONE)
                 (IM.TEDIT.DUMP.PARA))
                ((EQ IM.TEDIT.CR.FLG NIL)
                 (SETQ IM.TEDIT.CR.FLG 'ONE])
        (TAB (DUMP (CHARCODE TAB)))
        (FONT (CHANGE.FONT (CDR C))
              (push FONT.STACK (CDR C)))
        (INDENT (IM.ERROR "INDENT command encountered -- should be flushed"))
        (UNDO (SELECTQ (CDR C)
                   (FONT (SETQ FONT.STACK (CDR FONT.STACK))
                         (CHANGE.FONT (CAR FONT.STACK)))
                   (INDENT (IM.ERROR "UNDO INDENT command encountered -- should be flushed"))
                   NIL))
        (INVISIBLE                                                  (* print text <like index> which should be invisable, so it
                                                                    shouldn't start/stop paragraphs)
                    (PROG ((SAVE.CR.FLG IM.TEDIT.CR.FLG))
                          (DUMP (CDR C))
                          (SETQ IM.TEDIT.CR.FLG SAVE.CR.FLG)))
        (SHOULDNT])
```

(IM.TEDIT.DUMP.FOOTNOTES

```
  [LAMBDA NIL                                                       (* mjs " 4-Jun-85 15:44")
    (COND
       ([AND IM.TEDIT.FOOTNOTE.SAVES (NOT (GET.MY.PROP 'PASSFOOT))
             (NOT (GET.ANY.PARENT.PROP 'PASSFOOT]
        (PROG ((CURRENT.FOOTNOTES IM.TEDIT.FOOTNOTE.SAVES))
              (SETQ IM.TEDIT.FOOTNOTE.SAVES NIL)
              (for X in (REVERSE CURRENT.FOOTNOTES) do (IM.DUMP.CHARS X])
```

(IM.TEDIT.DUMP.PARA

```
  [LAMBDA NIL                                                       (* mjs " 4-Jun-85 15:46")
    (PROG NIL

          (* * actually end paragraph)

          (BOUT IM.OUTFILE (CHARCODE CR))

          (* * put out current paragraph formatting)

          (for X in (REVERSE IM.TEDIT.PARA.LOOKS) do (push IM.PARALOOKS X IM.TEDIT.LAST.PARA.BEGIN 1))

          (* * initialize vars for next paragraph)

          (SETQ IM.TEDIT.LAST.PARA.BEGIN (GETFILEPTR IM.OUTFILE))
          (SETQ IM.TEDIT.PARA.LOOKS NIL)
          (SETQ IM.TEDIT.CR.FLG 'MANY)

          (* * print out any footnotes waiting to be printed)

          (IM.TEDIT.DUMP.FOOTNOTES])
```

(INDEXX.PARSE.TYPE

```
  [LAMBDA (SAV)                                                     (* mjs "14-Jul-86 09:01")

          (* * Parse the type information from an INDEXX type field)

    (PROG (TYPE)
          (if (NULL SAV)
              then (RETURN 'TERM))
          (SETQ TYPE (PARSE.LIST SAV))

          (* if the type was specified with parenthesis at the beginning and the end, strip them out)

          [if (AND (LISTP TYPE)
                   (EQ (NTHCHARCODE (CAR TYPE)
                                    1)
                       (CHARCODE %())
                   (EQ (NTHCHARCODE (CAR (LAST TYPE))
                                    -1)
                       (CHARCODE %))))
              then (SETQ TYPE (CONS (SUBATOM (CAR TYPE)
                                             2 -1)
                                    (CDR TYPE)))
                   (SETQ TYPE (REVERSE (CONS (SUBATOM (CAR (REVERSE TYPE))
```

```
                                                    1 -2)
                                                (CDR (REVERSE TYPE]
            [if (TRANSLATE.SPECIAL.TYPES (CAR TYPE))
                then (SETQ TYPE (TRANSLATE.SPECIAL.TYPES (CAR TYPE]
            (RETURN (if TYPE
                        else 'TERM])
```

(FORMAT.DEF

```
  [LAMBDA (NAME TYPE SAV TYPESTRING)                                    (* mjs " 3-Oct-85 15:05")
```

(* * prints out a formatted definition. SAV should be a SAV-format text object which describes how the name/args should be formatted. NAME is the index-name inder which this definition should be grouped.
if SAV is NIL, NAME is used instead TYPE is the "object-type" of the defined object which is pased to the index.
TYPE is also printed in NIL after the function name. If TYPESTRING is given, it is used for TYPE in the printed definition, but TYPE is always used in the index.)

```
    (IM.HOLD.FOOTNOTES (DUMPOUT CR CR)
           (SEND.INFO (U-CASE NAME)
                  TYPE SAV '(*DEF*))
           (DUMPOUT PARALOOKS
                  `(QUAD LEFT 1STLEFTMARGIN %, IM.DEF.TITLE.1STLEFTMARGIN LEFTMARGIN %, IM.DEF.TITLE.LEFTMARGIN
                         LINELEADING 0 PARALEADING 18 POSTPARALEADING 0 TABS %, IM.RIGHT.MARGIN.TABS HEADINGKEEP
                         ON)
                  DUMP.CHARS SAV DUMP.CHARS "  " TAB FONT NIL DUMP.CHARS "[" DUMP.CHARS
                  (if TYPESTRING
                    else (LIST.TO.STRING TYPE))
                  DUMP.CHARS "]" CR CR)
           (if (EQ TO.ARG.NAME 'TEXT)
               then (DUMPOUT DUMP.CHARS (CREATE.VRULE.OBJECT T)))
           (DUMP.HRULE 1 NIL
                  `(QUAD LEFT 1STLEFTMARGIN %, IM.DEF.TITLE.1STLEFTMARGIN LEFTMARGIN %,
                         IM.DEF.TITLE.1STLEFTMARGIN LINELEADING 0 PARALEADING 0 POSTPARALEADING 0 HEADINGKEEP ON
                         ))
           (if (EQ TO.ARG.NAME 'TEXT)
               then (DUMPOUT CR CR PARALOOKS '(PARALEADING 0)
                             DUMP.ARG DUMP.CHARS (CREATE.VRULE.OBJECT)
                             CR CR)
                    (DUMP.HRULE 1 NIL)
                    (DUMPOUT CR CR PARALOOKS '(PARALEADING 18))
             elseif TO.ARG.NAME
               then (ERROR "FORMAT.DEF called when not at {TEXT or End of TO"])
```

(FORMAT.LISPWORD

```
  [LAMBDA (SAV)                                                         (* mjs " 2-MAY-83 18:08")
    (DUMPOUT FONT LISP DUMP.CHARS SAV])
```

(MAKE.IM.DOCUMENT

```
  [LAMBDA (FORM OUTFILE.FLG PAGE.LAYOUT OUTPUT.MESSAGE DEFAULT.PARALOOKS PTRFILENAME)
```

;; Edited 12-Apr-2024 19:58 by rmk

;; Edited 6-Mar-2024 21:17 by rmk: Fixed backquote commas. Also put IM.INDEX.CLOSEF calls in TEXTPROPs so advice in IMINDEX can be ;; eliminated.

;; Edited 20-Jul-2022 15:10 by rmk

(* mjs " 4-Aug-86 10:52")

;;; this function creates an IM output file, in XPS-compatible format. If sets up all of the special variables needed by DUMP, evaluates FORM, and sets ;;; all of the para and font looks

;;; If OUTFILE.FLG is NIL, the output file is just sent to the default printer. If OUTFILE.FLG is T, the outfile textstream is simply returned. If ;;; OUTFILE.FLG = anything else, it is taken as a file name to put the press file which is created <but not printed>.

;;; if PAGE.LAYOUT is non-NIL, it should be the compound page layout to be used.

;;; if OUTPUT.MESSAGE is non-NIL, it is printed on the hardcopy output

;;; PTRFILENAME is the name to be used if an index pointer file is generated during hardcopy <by printing index objects>

```
    (PROG ([IM.OUTFILE (OPENTEXTSTREAM NIL NIL NIL NIL `(IM.INDEX.PTRFILENAME ,PTRFILENAME]
           (FONT.STACK (CONS))
           (IM.TEDIT.LAST.PARA.BEGIN 1)
           (IM.TEDIT.LAST.FONT.BEGIN 1)
           (IM.TEDIT.PARA.LOOKS NIL)
           (IM.TEDIT.LAST.PARA.LOOKS NIL)
           (IM.TEDIT.FONT NIL)
           (IM.TEDIT.CR.FLG 'MANY)
           (IM.TEDIT.FOOTNOTE.SAVES NIL)
           (IM.TEDIT.SUB.SUPER.BEGIN NIL)
          IM.CHARLOOKS IM.PARALOOKS)
          (DECLARE (SPECVARS IM.OUTFILE FONT.STACK IM.TEDIT.LAST.PARA.BEGIN IM.TEDIT.LAST.FONT.BEGIN
                             IM.TEDIT.PARA.LOOKS IM.TEDIT.LAST.PARA.LOOKS IM.TEDIT.FONT IM.TEDIT.CR.FLG
                             IM.TEDIT.FOOTNOTE.SAVES IM.TEDIT.SUB.SUPER.BEGIN IM.CHARLOOKS IM.PARALOOKS))
          (SETQ IM.VRULE.STATE.LIST NIL)
          (DUMP.HEADERS.FOOTERS "  " "  "  ")
          (DUMPOUT CR CR START.PARA PARALOOKS `(TYPE PAGEHEADING SUBTYPE DRAFTMESSAGE QUAD LEFT 1STLEFTMARGIN 0
```

```
                                                    LEFTMARGIN 0 RIGHTMARGIN ,IM.TEXT.RIGHTMARGIN)
               DUMP.CHARS
               (COND
                  (IM.DRAFT.FLG (CONCAT "***DRAFT***  " (DATE)
                                        "  ***DRAFT***"))
                  (T "  "))
               CR CR)
         (EVAL FORM)
         (COND
            (IM.EVEN.FLG
                  ;; if you must quarantee that you have an even number of pages for two-sided copying, dump out a blank page no matter
                  ;; what -- it can always be discarded
                  (DUMPOUT CR CR START.PARA PARALOOKS `(NEWPAGEBEFORE T QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0
                                                       RIGHTMARGIN ,IM.TEXT.RIGHTMARGIN SPECIALX
                                                       ,IM.BLANKPAGE.SPECIALX SPECIALY
                                                       ,IM.BLANKPAGE.SPECIALY)
                        DUMP.CHARS "[This page intentionally left blank]" CR CR)))
```

;; after converting document, make sure that last para is formatted correctly by changing font, ending current para, and starting new para

```
(DUMPOUT CR CR FONT NIL)
(DUMP '(START.PARA))
```

;;; set page format

```
[TEDIT.PAGEFORMAT IM.OUTFILE
       (COND
          (PAGE.LAYOUT)
          (T (TEDIT.COMPOUND.PAGEFORMAT [TEDIT.SINGLE.PAGEFORMAT NIL NIL NIL NIL NIL
                                                 IM.PAGE.LEFTMARGIN IM.PAGE.RIGHTMARGIN
                                                 IM.PAGE.FIRST.TOPMARGIN IM.PAGE.BOTTOMMARGIN 1 NIL NIL
                                                 `((RECTOFOOT ,IM.PAGE.LEFTMARGIN ,IM.FOOTER.Y)
                                                   (RECTOFOOTRULE ,IM.PAGE.LEFTMARGIN ,IM.FOOTER.RULE.Y)
                                                   (DRAFTMESSAGE ,IM.DRAFT.MESSAGE.X
                                                                 ,IM.DRAFT.MESSAGE.BOTTOM.Y]
                                        [TEDIT.SINGLE.PAGEFORMAT NIL NIL NIL NIL NIL IM.PAGE.LEFTMARGIN
                                               IM.PAGE.RIGHTMARGIN IM.PAGE.TOPMARGIN IM.PAGE.BOTTOMMARGIN 1 NIL NIL
                                               `((DRAFTMESSAGE ,IM.DRAFT.MESSAGE.X ,IM.DRAFT.MESSAGE.TOP.Y)
                                                 (VERSOHEAD ,IM.PAGE.LEFTMARGIN ,IM.HEADER.Y)
                                                 (VERSOHEADRULE ,IM.PAGE.LEFTMARGIN ,IM.HEADER.RULE.Y)
                                                 (VERSOFOOT ,IM.PAGE.LEFTMARGIN ,IM.FOOTER.Y)
                                                 (VERSOFOOTRULE ,IM.PAGE.LEFTMARGIN ,IM.FOOTER.RULE.Y)
                                                 (DRAFTMESSAGE ,IM.DRAFT.MESSAGE.X ,IM.DRAFT.MESSAGE.BOTTOM.Y]
                                        (TEDIT.SINGLE.PAGEFORMAT NIL NIL NIL NIL NIL IM.PAGE.LEFTMARGIN
                                               IM.PAGE.RIGHTMARGIN IM.PAGE.TOPMARGIN IM.PAGE.BOTTOMMARGIN 1 NIL NIL
                                               `((DRAFTMESSAGE ,IM.DRAFT.MESSAGE.X ,IM.DRAFT.MESSAGE.TOP.Y)
                                                 (RECTOHEAD ,IM.PAGE.LEFTMARGIN ,IM.HEADER.Y)
                                                 (RECTOHEADRULE ,IM.PAGE.LEFTMARGIN ,IM.HEADER.RULE.Y)
                                                 (RECTOFOOT ,IM.PAGE.LEFTMARGIN ,IM.FOOTER.Y)
                                                 (RECTOFOOTRULE ,IM.PAGE.LEFTMARGIN ,IM.FOOTER.RULE.Y)
                                                 (DRAFTMESSAGE ,IM.DRAFT.MESSAGE.X ,IM.DRAFT.MESSAGE.BOTTOM.Y]
```

;;; dump default char and para looks for whole document -- and looks that should be different should be specified in the fns

```
(TEDIT.LOOKS IM.OUTFILE IM.TEXT.FONT 1 (GETFILEINFO IM.OUTFILE 'LENGTH))
(TEDIT.PARALOOKS IM.OUTFILE [COND
                               (DEFAULT.PARALOOKS)
                               (T `(QUAD JUSTIFIED 1STLEFTMARGIN ,IM.TEXT.LEFTMARGIN LEFTMARGIN
                                         ,IM.TEXT.LEFTMARGIN RIGHTMARGIN ,IM.TEXT.RIGHTMARGIN
                                         LINELEADING 0 PARALEADING 5 POSTPARALEADING 0]
       1
       (GETFILEINFO IM.OUTFILE 'LENGTH))
```

;; must reverse list because the order of some char and paragraph looks is significant << earlier looks are overridden by later ones >>

```
(SETQ IM.CHARLOOKS (DREVERSE IM.CHARLOOKS))
(SETQ IM.PARALOOKS (DREVERSE IM.PARALOOKS))
[while IM.CHARLOOKS bind (LOOKS CH# LEN) do (BLOCK)
                                            (SETQ LEN (pop IM.CHARLOOKS))
                                            (SETQ CH# (ADD1 (pop IM.CHARLOOKS)))
                                            (SETQ LOOKS (pop IM.CHARLOOKS))
                                            (COND
                                               ((IGREATERP LEN 0)
                                                (TEDIT.LOOKS IM.OUTFILE LOOKS CH# LEN]
[while IM.PARALOOKS bind (LOOKS CH# LEN) do (BLOCK)
                                            (SETQ LEN (pop IM.PARALOOKS))
                                            (SETQ CH# (ADD1 (pop IM.PARALOOKS)))
                                            (SETQ LOOKS (pop IM.PARALOOKS))
                                            (COND
                                               ((IGREATERP LEN 0)
                                                (TEDIT.PARALOOKS IM.OUTFILE LOOKS CH# LEN]
(COND
   ((NULL OUTFILE.FLG)
    (TEDIT.HARDCOPY IM.OUTFILE NIL NIL OUTPUT.MESSAGE)
    (TEDIT.KILL IM.OUTFILE)
    (printout T "Document sent to printer" T)
    (RETURN))
   ((EQ OUTFILE.FLG T)
```

```
             (RETURN IM.OUTFILE))
            (T (TEDIT.HARDCOPY IM.OUTFILE (PACKFILENAME 'BODY OUTFILE.FLG 'EXTENSION 'IP)
                    T OUTPUT.MESSAGE)
               (TEDIT.KILL IM.OUTFILE)
               (printout T "Output file: " (PACKFILENAME 'BODY OUTFILE.FLG 'EXTENSION 'IP)
                       T)
               (RETURN])
```

(PRINT.NOTE

```
  [LAMBDA (NOTE.STRING)                                          (* mjs "10-Apr-85 11:17")
    (COND
       (IM.NOTE.FLG (DUMPOUT FONT NOTE DUMP.CHARS "<<<" DUMP.CHARS NOTE.STRING DUMP.CHARS ">>>"])
```

(SEND.INFO

```
  [LAMBDA (NAME TYPE SAV INFO PLIST)                             (* mjs " 4-Aug-86 09:06")
    (if IM.INDEX.FILE.FLG
        then (DUMP.FORMAT 'INVISIBLE (IM.INDEX.CREATEOBJ (create IM.INDEX.DATA
                                                                 NAME _ NAME
                                                                 TYPE _ TYPE
                                                                 SAV _ (if (LISTP SAV)
                                                                           then (CAR SAV)
                                                                         else SAV)
                                                                 INFO _ INFO
                                                                 SUBSEC _ SUBSEC.COUNT.LIST
                                                                 PROPLIST _ PLIST])

)

;; fns for drawing vrules to the left of definition text -- an unused, never-fully debuged feature

(DEFINEQ

(IM.VRULE.DISPLAYFN
  [LAMBDA (OBJ STREAM)                                           (* mjs "19-Sep-85 15:06")
    (if (NOT (DISPLAYSTREAMP STREAM))
        then (push IM.VRULE.STATE.LIST (LIST (DSPYPOSITION NIL STREAM)
                                             (IMAGEOBJPROP OBJ 'IM.VRULE.STATE])

(CREATE.VRULE.OBJECT
  [LAMBDA (STATE)                                                (* mjs "11-Jul-86 12:17")
    (if IM.PRINT.VRULE.FLG
        then (PROG (OBJ)
                   [SETQ OBJ
                    (IMAGEOBJCREATE NIL
                           (if IM.VRULE.OBJECT.IMAGEFNS
                             else (SETQ IM.VRULE.OBJECT.IMAGEFNS
                                    (IMAGEFNSCREATE (FUNCTION IM.VRULE.DISPLAYFN)
                                           [FUNCTION (LAMBDA NIL
                                                      (create IMAGEBOX
                                                             XSIZE _ 0
                                                             YSIZE _ 0
                                                             YDESC _ 0
                                                             XKERN _ 0]
                                           'NILL
                                           (FUNCTION CREATE.VRULE.OBJECT)
                                           'NILL
                                           'NILL
                                           'NILL
                                           'NILL
                                           'NILL
                                           'NILL
                                           'NILL
                                           'NILL
                                           'NILL
                                           'IM.VRULE.OBJECT]
                   (IMAGEOBJPROP OBJ 'IM.VRULE.STATE STATE)
                   (RETURN OBJ))
      else ""])

(PRINT.VRULES.ON.PAGE
  [LAMBDA (STREAM)                                               (* mjs "23-Sep-85 14:14")
    (COND
       ((AND IM.PRINT.VRULE.FLG (NOT (DISPLAYSTREAMP STREAM)))
        (for X in (REVERSE (SORT IM.VRULE.STATE.LIST T)) bind (SCALED.VRULE.WIDTH _ (TIMES 1 (DSPSCALE NIL STREAM)
                                                                                            ))
                                                              (SCALED.VRULE.X _ (TIMES (PLUS IM.VRULE.X
                                                                                             IM.PAGE.LEFTMARGIN)
                                                                                       (DSPSCALE NIL STREAM)))
                                                              (STATE _ NIL)
                                                              (YPOS _ NIL)
                                                              CURRENT.STATE CURRENT.YPOS
           do (SETQ CURRENT.STATE (CADR X))
              (SETQ CURRENT.YPOS (CAR X))
```

```
            (COND
               ((AND (NULL CURRENT.STATE)
                     STATE
                     (NUMBERP YPOS)
                     (NUMBERP CURRENT.YPOS))
                (DRAWLINE SCALED.VRULE.X YPOS SCALED.VRULE.X CURRENT.YPOS SCALED.VRULE.WIDTH NIL STREAM)))
            (COND
               (CURRENT.STATE (SETQ STATE CURRENT.STATE)
                      (SETQ YPOS CURRENT.YPOS))
               (T (SETQ STATE NIL)
                  (SETQ YPOS NIL)))
          finally (PROGN (SETQ IM.VRULE.STATE.LIST NIL)
                         (COND
                            ((AND STATE (NUMBERP YPOS))
                             (DRAWLINE SCALED.VRULE.X YPOS SCALED.VRULE.X (TIMES IM.TEXT.BOTTOMMARGIN
                                                                                 (DSPSCALE NIL STREAM))
                                   SCALED.VRULE.WIDTH NIL STREAM)
                             (push IM.VRULE.STATE.LIST (LIST (TIMES IM.TEXT.TOPMARGIN (DSPSCALE NIL STREAM))
                                                             STATE])

)

(RPAQQ IM.VRULE.STATE.LIST NIL)

(RPAQ? IM.VRULE.OBJECT.IMAGEFNS NIL)

(RPAQ? IM.PRINT.VRULE.FLG NIL)
```

;; fns for printing page numbers

```
(DEFINEQ
```

(IM.FOLIO.DISPLAYFN

```
  [LAMBDA (OBJ STREAM)                                    (* mjs "20-Sep-85 09:45")
    (PROG ((FOLIO.TEXT (GET.FOLIO.STRING OBJ STREAM))
           FONT OFONT)
          (SETQ FONT (SELECTQ (IMAGESTREAMTYPE STREAM)
                              (DISPLAY '(MODERN 8 MRR 0 DISPLAY))
                              (PRESS '(MODERN 8 MRR 0 PRESS))
                              (INTERPRESS '(MODERN 8 MRR 0 INTERPRESS))
                              NIL))
          (SETQ OFONT (DSPFONT FONT STREAM))
          (PRIN1 FOLIO.TEXT STREAM)
          (DSPFONT OFONT STREAM)

          (* * take care of any vrules on page)

          (PRINT.VRULES.ON.PAGE STREAM])
```

(IM.FOLIO.SIZEFN

```
  [LAMBDA (OBJ STREAM CURX RIGHTMARGIN)                   (* mjs " 7-Aug-85 10:27")
                                                          (* Tell the size of a menu button)
    (PROG ((FOLIO.TEXT (GET.FOLIO.STRING OBJ STREAM))
           FONT BOX)
          (SETQ FONT (SELECTQ (IMAGESTREAMTYPE STREAM)
                              (DISPLAY '(MODERN 8 MRR 0 DISPLAY))
                              (PRESS '(MODERN 8 MRR 0 PRESS))
                              (INTERPRESS '(MODERN 8 MRR 0 INTERPRESS))
                              NIL))
          (SETQ BOX (create IMAGEBOX
                            XSIZE _ (STRINGWIDTH FOLIO.TEXT FONT)
                            YSIZE _ (FONTPROP FONT 'HEIGHT)
                            YDESC _ (FONTPROP FONT 'DESCENT)
                            XKERN _ 0))
          (RETURN BOX])
```

(CREATE.FOLIO.OBJECT

```
  [LAMBDA NIL                                             (* mjs "11-Jul-86 12:16")
    (PROG (OBJ)
          [SETQ OBJ (IMAGEOBJCREATE NIL (COND
                                          (IM.FOLIO.OBJECT.IMAGEFNS)
                                          (T (SETQ IM.FOLIO.OBJECT.IMAGEFNS (IMAGEFNSCREATE
                                                                              (FUNCTION IM.FOLIO.DISPLAYFN)
                                                                              (FUNCTION IM.FOLIO.SIZEFN)
                                                                              'NILL
                                                                              (FUNCTION CREATE.FOLIO.OBJECT)
                                                                              'NILL
                                                                              'NILL
                                                                              'NILL
                                                                              'NILL
                                                                              'NILL
                                                                              'NILL
                                                                              'NILL
                                                                              'NILL
                                                                              'NILL
```

```
                                                                    'IM.FOLIO.OBJECT]
        (IMAGEOBJPROP OBJ 'IM.CHAPNUM.DATA (COND
                                             ((BOUNDP 'SUBSEC.COUNT.LIST)
                                              (CAR (LAST SUBSEC.COUNT.LIST)))
                                             (T NIL)))
        (RETURN OBJ])
```

(GET.FOLIO.STRING

```
  [LAMBDA (OBJ STREAM)                                          (* mjs "19-Sep-85 14:59")
    (PROG ([CHAPNUM.INFO (MKLIST (IMAGEOBJPROP OBJ 'IM.CHAPNUM.DATA]
           CHAPNUM PAGENUM)
          (SETQ CHAPNUM (CAR CHAPNUM.INFO))
          [SETQ PAGENUM (COND
                           ((DISPLAYSTREAMP STREAM)
                            "xx")
                           (T (PROG [(TEDIT.PAGE (fetch (PAGEFORMATTINGSTATE PAGE#) of FORMATTINGSTATE))
                                     (NUMBER.PAGENUMS (LENGTH (CDR CHAPNUM.INFO]
                                    (RETURN (COND
                                               ((NULL (CDR CHAPNUM.INFO))
                                                TEDIT.PAGE)
                                               [(GREATERP TEDIT.PAGE NUMBER.PAGENUMS)
                                                (COND
                                                   ((NUMBERP (CAR (LAST CHAPNUM.INFO)))
                                                    (IPLUS (CAR (LAST CHAPNUM.INFO))
                                                           (IDIFFERENCE TEDIT.PAGE NUMBER.PAGENUMS)))
                                                   (T (CAR (LAST CHAPNUM.INFO]
                                               (T (CAR (NTH (CDR CHAPNUM.INFO)
                                                            TEDIT.PAGE]
          (RETURN (COND
                     (CHAPNUM (CONCAT CHAPNUM "." PAGENUM))
                     (T (MKSTRING PAGENUM])

)

(RPAQ? IM.FOLIO.OBJECT.IMAGEFNS NIL)
```

;; TOPROG functions, used to define the translating actions of IM text objects.

```
(DEFINEQ
```

(ARG#TOPROG

```
  [LAMBDA NIL                                                   (* mjs "14-APR-83 16:10")
    (DUMPOUT FONT ARG DUMP.ARG])
```

(BIGLISPCODE#TOPROG

```
  [LAMBDA NIL                                                   (* mjs " 8-Aug-85 09:24")
    (DUMPOUT CR CR)
    (IM.HOLD.FOOTNOTES (PROG [(SAV (SAVE.ARG))
                              (NEW.LINE (CONS))
                              (LISP.LINE.PARA.LOOKS '(QUAD LEFT LEFTMARGIN 0 1STLEFTMARGIN 0]
                             (TCONC SAV (CHARCODE CR))
                             (for X in (CAR SAV)
                                do (TCONC NEW.LINE X)
                                   (if (EQ X (CHARCODE CR))
                                       then (DUMPOUT FONT LISP PARALOOKS LISP.LINE.PARA.LOOKS START.PARA
                                                     DUMP.CHARS NEW.LINE CR CR)
                                                                  (* after first line, use 0 para leading)
                                            (SETQ LISP.LINE.PARA.LOOKS
                                             '(QUAD LEFT PARALEADING 0 LEFTMARGIN 0 1STLEFTMARGIN 0))
                                            (SETQ NEW.LINE (CONS])
```

(BRACKET#TOPROG

```
  [LAMBDA NIL                                                   (* mjs "11-APR-83 11:10")
    (DUMPOUT DUMP.CHARS "{" DUMP.ARG DUMP.CHARS "}"])
```

(CHAPTER#TOPROG

```
  [LAMBDA NIL                                                   (* mjs "18-Sep-85 14:54")
    (SAVE.ARGS NUMBER TITLE)
    (PROG ((TITLE.SAV (GET.ARG.SAV TITLE))
           (CHAP.NUM (PARSE.ATOM (GET.ARG.SAV NUMBER)))
           UCASE.TITLE.STRING CHAP.NUM.STRING)
          (SETQ UCASE.TITLE.STRING (U-CASE (PARSE.STRING TITLE.SAV)))
          [if (NULL CHAP.NUM)
              then (SETQ CHAP.NUM (if (BOUNDP 'GLOBAL.CHAPTER.NUMBER)
                                      then GLOBAL.CHAPTER.NUMBER
                                     else (printout T "No number specified for chapter: '" UCASE.TITLE.STRING
                                                  "' --- please type chapter number: ")
                                          (READ]
          (SETQ SUBSEC.COUNT.LIST (CONS CHAP.NUM))
          (SETQ SUBSEC.LAST.SUB 0)
          (SETQ FOOTNOTE.NUM 0)
          (SETQ FIGURE.NUM 0)
```

```
        (* specify both headers and footers, in case a page break comes before the next subsec)

        (DUMP.HEADERS.FOOTERS UCASE.TITLE.STRING UCASE.TITLE.STRING)
        (SEND.INFO (MKATOM UCASE.TITLE.STRING)
               'CHAPTER TITLE.SAV)
        (SETQ CHAP.NUM (MKLIST CHAP.NUM))
        (SETQ CHAP.NUM.STRING (if (CAR CHAP.NUM)
                                  then (CONCAT (CAR CHAP.NUM)
                                               ".")
                                  else ""))
        (DUMPOUT FONT IM.CHAPTER.TITLE.FONT PARALOOKS `(PARALEADING 0 LINELEADING 0 QUAD LEFT TABS %,
                                                                  IM.RIGHT.MARGIN.TABS)
               DUMP.CHARS CHAP.NUM.STRING TAB DUMP.CHARS UCASE.TITLE.STRING CR CR)

        (* we want the PARALEADING between the chapter rule and the next line to be 108pts.
        ASSUMING that the next line is a normal text line, rather than a subsec heading, this means that the next para will have a
        paraleading of 5pts. Therefore, this "invisible" paragraph must take 103pts.
        If the font size is 10pts and the paraleading is 93pts, this should do the trick.)

        (DUMP.HRULE 6)
        (DUMPOUT START.PARA FONT NIL PARALOOKS '(PARALEADING 93 LINELEADING 0)
               DUMP.CHARS " " CR CR)

        (* theoretically, we should be able to get the space we need by using POSTPARALEADING, but this doesn't seem to work.
        Try%: (DUMP.HRULE 6 NIL (QUOTE (POSTPARALEADING 103))))

        (if (EQ TO.ARG.NAME 'TEXT)
            then (DUMPOUT DUMP.ARG CR CR])
```

(COMMENT#TOPROG

```
  [LAMBDA NIL                                                    (* mjs " 6-Aug-85 15:15")
    (PROG ((IM.INDEX.FILE.FLG NIL)
           (IM.REF.FLG NIL)
           (IM.SEND.IMPLICIT NIL)
           (IM.CHECK.DEFS NIL))
          (DECLARE (SPECVARS IM.INDEX.FILE.FLG IM.REF.FLG IM.SEND.IMPLICIT IM.CHECK.DEFS))
                                                                 (* make sure that no ptrs are sent from TOs in a note or
                                                                 comment)
          (FLUSH.ARG])
```

(DEF#TOPROG

```
  [LAMBDA NIL                                                    (* mjs " 9-Apr-85 16:00")
    (SAVE.ARGS TYPE NAME PRINTNAME ARGS PARENS NOPARENS)
    (PROG [[PARENS.FLG (OR (GET.ARG.SAV PARENS)
                           (AND (GET.ARG.SAV NAME)
                                (GET.ARG.SAV ARGS)
                                (NULL (GET.ARG.SAV NOPARENS]
           [INDEX.NAME (MKATOM (LIST.TO.STRING (PARSE.LIST (GET.ARG.SAV NAME]
           (PARSED.TYPE (PARSE.LIST (GET.ARG.SAV TYPE]           (* if no more args, just return)
          (if (TRANSLATE.SPECIAL.TYPES (CAR PARSED.TYPE))
              then (SETQ PARSED.TYPE (TRANSLATE.SPECIAL.TYPES (CAR PARSED.TYPE)))
            elseif (AND (EQ (NTHCHAR (CAR PARSED.TYPE)
                                     1)
                            '%(
                        (EQ (NTHCHAR (CAR (LAST PARSED.TYPE))
                                     -1)
                            '%)))
                 then (SETQ PARSED.TYPE (CAR (GET.ARG.SAV TYPE)))
                      (SETQ PARSED.TYPE (CDR (MEMB (CHARCODE %()
                                                   PARSED.TYPE)))
                      [SETQ PARSED.TYPE (REVERSE (CDR (MEMB (CHARCODE %))
                                                            (REVERSE PARSED.TYPE]
                      [SETQ PARSED.TYPE (PARSE.LIST (CONS PARSED.TYPE (LAST PARSED.TYPE]
            else (IM.ERROR "bad TYPE arg given to DEF of " INDEX.NAME "  '" (PARSE.STRING PARSED.TYPE)
                           "'  ---  TERM used instead")
                 (SETQ PARSED.TYPE 'TERM))
          (FORMAT.DEF INDEX.NAME PARSED.TYPE
                 (if (GET.ARG.SAV PRINTNAME)
                     then (SAVE.DUMPOUT FONT LISP DUMP.CHARS (GET.ARG.SAV PRINTNAME))
                   else (if (GET.ARG.SAV ARGS)
                            then (if PARENS.FLG
                                     then (SAVE.DUMPOUT FONT LISP DUMP.CHARS "(" DUMP.CHARS (GET.ARG.SAV NAME)
                                                        DUMP.CHARS " " FONT ARG DUMP.CHARS (GET.ARG.SAV ARGS)
                                                        FONT LISP DUMP.CHARS ")")
                                   else (SAVE.DUMPOUT FONT LISP DUMP.CHARS (GET.ARG.SAV NAME)
                                                      DUMP.CHARS " " FONT ARG DUMP.CHARS (GET.ARG.SAV ARGS)))
                          else (if PARENS.FLG
                                   then (SAVE.DUMPOUT FONT LISP DUMP.CHARS "(" DUMP.CHARS (GET.ARG.SAV NAME)
                                                      DUMP.CHARS ")")
                                 else (SAVE.DUMPOUT FONT LISP DUMP.CHARS (GET.ARG.SAV NAME])
```

(FIGURE#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "25-Jul-85 11:30")
    (PROG ((DUMP.CAPTION.FLG NIL)
```

```
           (CHAP.NUM (CAR (LAST SUBSEC.COUNT.LIST)))
           (FIGURE.TAG.LIST NIL)
           FIGURE.TAG ARG.NAME FIGURE.NUM.STRING)
          (SETQ FIGURE.NUM (ADD1 FIGURE.NUM))
          (SETQ FIGURE.NUM.STRING (if (EQ CHAP.NUM 0)
                                      then (CONCAT "Figure " FIGURE.NUM ".")
                                    else (CONCAT "Figure " CHAP.NUM "." FIGURE.NUM ".")))
          (DUMPOUT CR CR)
      loop
          (SELECTQ (SETQ ARG.NAME (GET.ARG))
              (TAG [SETQ FIGURE.TAG (U-CASE (PARSE.ATOM (SAVE.ARG]
                   (SETQ FIGURE.TAG.LIST (CONS FIGURE.TAG FIGURE.TAG.LIST))
                   (SEND.INFO FIGURE.TAG 'TAG NIL (LIST '*FIGURE* FIGURE.NUM)))
              (TEXT (DUMPOUT DUMP.ARG CR CR))
              (CAPTION (DUMPOUT DUMP.CHARS FIGURE.NUM.STRING DUMP.CHARS "   " DUMP.ARG CR CR)
                       (SETQ DUMP.CAPTION.FLG T))
              (NIL (if (NOT DUMP.CAPTION.FLG)
                       then (DUMPOUT DUMP.CHARS FIGURE.NUM.STRING CR CR))
                   (SAVE.INFILE.NOTE 'IM.FIGURE (CONS FIGURE.NUM FIGURE.TAG.LIST))
                   (RETURN))
              (SHOULDNT))
          (GO loop])
```

(FN#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "13-SEP-83 17:14")
    (PROG ((SAV (SAVE.ARG)))
          (FORMAT.LISPWORD SAV)
          (SEND.IMPLICIT (PARSE.ATOM SAV)
                 '(Function)
                 SAV)
          (if [AND IM.CHECK.DEFS (NOT (GETD (PARSE.ATOM SAV]
              then (SAVE.INFILE.NOTE 'UNDEF.FN (PARSE.ATOM SAV])
```

(FNDEF#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "18-Sep-85 14:57")
    (SAVE.ARGS NAME ARGS TYPE)
    (PROG ((NAME (PARSE.ATOM (GET.ARG.SAV NAME)))
           (ARGS (PARSE.LIST (GET.ARG.SAV ARGS)))
           [TYPES (U-CASE (PARSE.LIST (GET.ARG.SAV TYPE]
           NEXT.ARG typestring fntype typelist)                  (* if no more args, just return)
          (DUMPOUT CR CR)
          (if IM.CHECK.DEFS
              then (if (GETD NAME)
                       then (SETQ fntype 0)
                            (COND
                               ((FMEMB 'NLAMBDA TYPES)
                                (SETQ fntype 1)))
                            [COND
                               ((FMEMB 'NOSPREAD TYPES)
                                (SETQ fntype (IPLUS fntype 2]
                            (COND
                               ((NEQ fntype (ARGTYPE NAME))
                                (SETQ typelist (SELECTQ (ARGTYPE NAME)
                                                        (0 '[LAMBDA SPREAD])
                                                        (1 '[NLAMBDA SPREAD])
                                                        (2 '[LAMBDA NOSPREAD])
                                                        (3 '[NLAMBDA NOSPREAD])
                                                        NIL))
                                (DUMPOUT CR CR)
                                (PRINT.NOTE (CONCAT NAME " is a " typelist " but manual def says it is a " TYPES)
                                            )
                                (IM.WARNING NAME " is a " typelist " but manual def says it is a " TYPES)))
                            (COND
                               ([NOT (OR (EQUAL ARGS (MKLIST (ARGLIST NAME)))
                                         (AND (NLISTP (ARGLIST NAME))
                                              (GREATERP (LENGTH ARGS)
                                                        1]
                                (DUMPOUT CR CR)
                                (PRINT.NOTE (CONCAT NAME " has arglist: " (MKLIST (ARGLIST NAME))
                                                    " in Interlisp-D"))
                                (IM.WARNING NAME " has arglist: " (MKLIST (ARGLIST NAME)))
                                (IM.WARNING "      but manual says: " ARGS)))
                     else (DUMPOUT CR CR)
                          (PRINT.NOTE (CONCAT "Function: " NAME " is not defined in Interlisp-D"))
                          (IM.WARNING "Function: " NAME " is not defined in Interlisp-D")
                          (SAVE.INFILE.NOTE 'UNDEF.FN NAME)))
          (DUMPOUT CR CR)
          (SETQ typestring "Function")
          [COND
             ((FMEMB 'NOSPREAD TYPES)
              (SETQ typestring (CONCAT "NoSpread " typestring]
          [COND
             ((FMEMB 'NLAMBDA TYPES)
              (SETQ typestring (CONCAT "NLambda " typestring]
          (FORMAT.DEF NAME '(Function)
```

```
                (if (AND (LISTP (GET.ARG.SAV ARGS))
                         (CAR (GET.ARG.SAV ARGS)))
                    then (SAVE.DUMPOUT FONT LISP DUMP.CHARS "(" DUMP.CHARS (GET.ARG.SAV NAME)
                                DUMP.CHARS " " FONT ARG DUMP.CHARS (GET.ARG.SAV ARGS)
                                FONT LISP DUMP.CHARS ")")
                  else (SAVE.DUMPOUT FONT LISP DUMP.CHARS "(" DUMP.CHARS (GET.ARG.SAV NAME)
                              DUMP.CHARS ")"))
                typestring])
```

(FOOT#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "18-Sep-85 14:57")
    (PROG (FOOT.SAV)
          (SETQ FOOTNOTE.NUM (ADD1 FOOTNOTE.NUM))
          (IM.WARNING "footnote #" FOOTNOTE.NUM "  --- not fully implemented")
          (DUMPOUT FONT FOOTNOTE START.SUPER DUMP.CHARS FOOTNOTE.NUM END.SUPER)
          (SETQ FOOT.SAV (SAVE.DUMPOUT FONT FOOTNOTE START.SUPER DUMP.CHARS FOOTNOTE.NUM END.SUPER DUMP.ARG CR
                                       CR))
          (push IM.TEDIT.FOOTNOTE.SAVES FOOT.SAV])
```

(INCLUDE#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "25-Jul-85 10:49")
    (PROG [(names (PARSE.LIST (SAVE.ARG]
          (COND
             ((CDR names)
              (IM.ERROR "Include file name: " names " -- first name will be used")))
          (INCLUDE.FILE (CAR names])
```

(INDEX#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "23-Jul-85 11:21")
    (PROG ((SAV (SAVE.ARG))
           (INFO NIL)
           TYPE ARGS TEMP ARG.ATOM)
          (SETQ TEMP (PARSE.INDEX.SPEC SAV T))
          (if (OR (NULL TEMP)
                  (NULL (CAR TEMP)))
              then (IM.WARNING "null index with type=" (CDR TEMP)
                               " --- ignored")
                   (RETURN))
          (SETQ ARGS (CAR TEMP))
          (SETQ TYPE (CDR TEMP))
          (while (FMEMB (CAR ARGS)
                        '(*BEGIN* *END* *PRIMARY*))
             do (SETQ INFO (CONS (CAR ARGS)
                                 INFO))
                (SETQ ARGS (CDR ARGS)))
          (SETQ ARG.ATOM (MKATOM (LIST.TO.STRING ARGS)))
          (if (U-CASEP ARG.ATOM)
              then (SEND.INFO ARG.ATOM TYPE NIL INFO)
            else (SEND.INFO (U-CASE ARG.ATOM)
                            TYPE ARG.ATOM INFO])
```

(INDEXX#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "11-Jul-86 16:57")
    (SAVE.ARGS NAME TYPE INFO TEXT SUBNAME SUBTYPE SUBTEXT SUBSUBNAME SUBSUBTYPE SUBSUBTEXT)
    (PROG ((PROPLIST NIL)
           [NAME (MKATOM (LIST.TO.STRING (if (PARSE.LIST (GET.ARG.SAV NAME))
                                             else '(NIL]
           (TYPE (INDEXX.PARSE.TYPE (GET.ARG.SAV TYPE)))
           (INFO (PARSE.LIST (GET.ARG.SAV INFO)))
           [SUBNAME (MKATOM (LIST.TO.STRING (if (PARSE.LIST (GET.ARG.SAV SUBNAME))
                                                else '(NIL]
           (SUBTYPE (INDEXX.PARSE.TYPE (GET.ARG.SAV SUBTYPE)))
           [SUBSUBNAME (MKATOM (LIST.TO.STRING (if (PARSE.LIST (GET.ARG.SAV SUBSUBNAME))
                                                   else '(NIL]
           (SUBSUBTYPE (INDEXX.PARSE.TYPE (GET.ARG.SAV SUBSUBTYPE)))
           TEXT SUBTEXT SUBSUBTEXT)
          (SETQ TEXT (if (GET.ARG.SAV TEXT)
                         elseif (NOT (U-CASEP NAME))
                           then NAME))
          (SETQ SUBTEXT (if (GET.ARG.SAV SUBTEXT)
                            elseif (NOT (U-CASEP SUBNAME))
                              then SUBNAME))
          (SETQ SUBSUBTEXT (if (GET.ARG.SAV SUBSUBTEXT)
                               elseif (NOT (U-CASEP SUBSUBNAME))
                                 then SUBSUBNAME))
          (SETQ NAME (U-CASE NAME))
          (SETQ SUBNAME (U-CASE SUBNAME))
          (SETQ SUBSUBNAME (U-CASE SUBSUBNAME))
          [if (OR SUBNAME SUBTEXT)
              then (SETQ PROPLIST (APPEND PROPLIST (LIST 'SUBNAME SUBNAME 'SUBTYPE SUBTYPE 'SUBTEXT SUBTEXT]
          [if (OR SUBSUBNAME SUBSUBTEXT)
              then (SETQ PROPLIST (APPEND PROPLIST (LIST 'SUBSUBNAME SUBSUBNAME 'SUBSUBTYPE SUBSUBTYPE
                                                         'SUBSUBTEXT SUBSUBTEXT]
```

```
          (SEND.INFO NAME TYPE TEXT INFO PROPLIST])
```

(IT#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "18-APR-83 14:32")
    (DUMPOUT FONT ITALIC DUMP.ARG])
```

(LBRACKET#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "10-Apr-85 09:51")
    (IM.DUMP.CHARS "{")
    (TRIVIAL.ARG])
```

(LISP#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "18-APR-83 14:27")
    (DUMPOUT FONT LISP DUMP.ARG])
```

(LISPCODE#TOPROG

```
  [LAMBDA NIL                                                    (* mjs " 2-Aug-85 16:27")
    (DUMPOUT CR CR)
    (IM.HOLD.FOOTNOTES (PROG [(SAV (SAVE.ARG))
                              (NEW.LINE (CONS))
                              (LISP.LINE.PARA.LOOKS '(QUAD LEFT]
                             (TCONC SAV (CHARCODE CR))
                             (for X in (CAR SAV)
                                do (TCONC NEW.LINE X)
                                   (if (EQ X (CHARCODE CR))
                                       then (DUMPOUT FONT LISP PARALOOKS LISP.LINE.PARA.LOOKS START.PARA
                                                     DUMP.CHARS NEW.LINE CR CR)
                                                                  (* after first line, use 0 para leading)
                                            (SETQ LISP.LINE.PARA.LOOKS '(QUAD LEFT PARALEADING 0))
                                            (SETQ NEW.LINE (CONS])
```

(LISPWORD#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "27-JUL-83 14:13")

          (* keep as seperate fn from LISP#TOPROG so can easily add hacks to check fns, etc..)

    (PROG ((SAV (SAVE.ARG)))
          (FORMAT.LISPWORD SAV)
          (SEND.IMPLICIT (PARSE.ATOM SAV)
                 (SELECTQ TO.NAME
                      (ATOM '(Litatom))
                      (BREAKCOM '(Break Command))
                      (EDITCOM '(Editor Command))
                      (FILECOM '(File Package Command))
                      (MAC '(Macro))
                      (PACOM '(Prog. Asst. Command))
                      (PROP '(Property Name))
                      'TERM)
                 SAV])
```

(LIST#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "28-Jul-86 16:36")
    (PROG ((LNAME.AND.NAME.LIST NIL)
           (ITEM.NUMBER 0)
           (LIST.PARA.LOOKS `(1STLEFTMARGIN 0 LEFTMARGIN %, IM.TEXT.LEFTMARGIN POSTPARALEADING 0 TABS %,
                                    IM.LABELED.LIST.TABS))
           LAST.SPEC)

          (* * LNAME.AND.NAME.LIST is a list of LNAME and NAME values, where each element of the list is of the form
          (LNAME . NAME))

          (DUMPOUT CR CR)
          (until (NULL (SETQ LAST.SPEC (GET.ARG)))
             do (SELECTQ LAST.SPEC
                      ((INDENT MAX)
                           (IM.WARNING "List with " LAST.SPEC (PARSE.NUMBER.OR.PERCENTAGE (SAVE.ARG)
                                                                                          100 100)
                                       " spec -- de-implemented"))
                      (UNINDENTED (DUMPOUT CR CR DUMP.ARG CR CR))
                      (LNAME (push LNAME.AND.NAME.LIST (CONS (SAVE.ARG)
                                                             " ")))
                      (NAME [if [AND LNAME.AND.NAME.LIST (EQUAL " " (CDR (CAR LNAME.AND.NAME.LIST]
                                then (RPLACD (CAR LNAME.AND.NAME.LIST)
                                             (SAVE.ARG))
                              else (push LNAME.AND.NAME.LIST (CONS " " (SAVE.ARG])
                      (ITEM (SETQ ITEM.NUMBER (ADD1 ITEM.NUMBER))
                            (SELECTQ TO.NAME
                                 (NUMBEREDLIST (DUMPOUT PARALOOKS LIST.PARA.LOOKS TAB DUMP.CHARS "(" DUMP.CHARS
                                                        ITEM.NUMBER DUMP.CHARS ")" TAB DUMP.ARG CR CR))
                                 (UNNUMBEREDLIST
                                      (DUMPOUT PARALOOKS LIST.PARA.LOOKS TAB DUMP.CHARS (MKSTRING
```

```
                                                                    (CHARACTER (CHARCODE
                                                                                %#7)))
                                        TAB DUMP.ARG CR CR))
                                 (LABELEDLIST (if LNAME.AND.NAME.LIST
                                                  then (PROG ((ALL.BUT.LAST.NAME.SPECS (REVERSE (CDR
                                                                                                LNAME.AND.NAME.LIST
                                                                                                     )))
                                                              (FIRST.NAME.SPEC (CAR LNAME.AND.NAME.LIST)))
                                                             [for LNAME.NAME in ALL.BUT.LAST.NAME.SPECS
                                                                do
                                                                   (* dump all but last name)
                                                                   (DUMPOUT PARALOOKS LIST.PARA.LOOKS DUMP.CHARS
                                                                          (CAR LNAME.NAME)
                                                                          TAB PARALOOKS '(HEADINGKEEP ON)
                                                                          DUMP.CHARS
                                                                          (CDR LNAME.NAME)
                                                                          TAB CR CR PARALOOKS '(PARALEADING 0]
                                                                   (* dump last name and item)
                                                             (DUMPOUT PARALOOKS LIST.PARA.LOOKS DUMP.CHARS
                                                                    (CAR FIRST.NAME.SPEC)
                                                                    TAB DUMP.CHARS (CDR FIRST.NAME.SPEC)
                                                                    TAB DUMP.ARG CR CR))
                                                  else (DUMPOUT DUMP.ARG CR CR))
                                                (SETQ LNAME.AND.NAME.LIST NIL))
                                 (SHOULDNT)))
                          (SHOULDNT])
```

(MACDEF#TOPROG

```
[LAMBDA NIL                                                    (* mjs " 5-AUG-83 13:31")
  (SAVE.ARGS NAME ARGS TYPE)
  (PROG ((NAME (PARSE.ATOM (GET.ARG.SAV NAME)))
         (ARGS (PARSE.LIST (GET.ARG.SAV ARGS)))
         [TYPES (U-CASE (PARSE.LIST (GET.ARG.SAV TYPE]
         typestring)

        (* * will eventually check if NAME has a macro definition)

        (SETQ typestring "Macro")
        [COND
           ((FMEMB 'NOSPREAD TYPES)
            (SETQ typestring (CONCAT "NoSpread " typestring]
        [COND
           ((FMEMB 'NLAMBDA TYPES)
            (SETQ typestring (CONCAT "NLambda " typestring]
        (FORMAT.DEF NAME '(Macro)
               (if (GET.ARG.SAV ARGS)
                   then (SAVE.DUMPOUT FONT LISP DUMP.CHARS "(" DUMP.CHARS (GET.ARG.SAV NAME)
                               DUMP.CHARS " " FONT ARG DUMP.CHARS (GET.ARG.SAV ARGS)
                               FONT LISP DUMP.CHARS ")")
                 else (SAVE.DUMPOUT FONT LISP DUMP.CHARS "(" DUMP.CHARS (GET.ARG.SAV NAME)
                             DUMP.CHARS ")"))
             typestring])
```

(NOTE#TOPROG

```
[LAMBDA NIL                                                    (* mjs " 6-Aug-85 15:14")
  (PROG ((IM.INDEX.FILE.FLG NIL)
         (IM.REF.FLG NIL)
         (IM.SEND.IMPLICIT NIL)
         (IM.CHECK.DEFS NIL))
        (DECLARE (SPECVARS IM.INDEX.FILE.FLG IM.REF.FLG IM.SEND.IMPLICIT IM.CHECK.DEFS))
                                                               (* make sure that no ptrs are sent from TOs in a note or
                                                               comment)
        (if IM.NOTE.FLG
            then (DUMPOUT FONT NOTE DUMP.CHARS "<<<< " DUMP.ARG DUMP.CHARS " >>>>")
          else (FLUSH.ARG])
```

(PRINT.SPECIAL.CHARS#TOPROG

```
[LAMBDA NIL                                                    (* mjs " 4-Oct-85 13:45")
  (PROG [(CHAR.STRING (SELECTQ TO.NAME
                           (ANONARG (MKSTRING (CHARACTER (CHARCODE 357,45))))
                           (BULLET (MKSTRING (CHARACTER (CHARCODE %#7))))
                           (CRSYMBOL (DUMPOUT START.SUPER DUMP.CHARS "cr" END.SUPER)
                                     "")
                           (ELLIPSIS "...")
                           (EMDASH (MKSTRING (CHARACTER (CHARCODE 357,45))))
                           (ENDASH (MKSTRING (CHARACTER (CHARCODE 357,44))))
                           (GE ">=")
                           (LE "<=")
                           (NE "~=")
                           (PI "~PI~")
                           (PLUSMINUS "+-")
                           (SP " ")
                           (SHOULDNT]
        (DUMPOUT DUMP.CHARS CHAR.STRING TRIVIAL.ARG])
```

```
(PROPDEF#TOPROG
  [LAMBDA NIL                                                    (* mjs " 5-MAY-83 11:56")
    (SAVE.ARGS NAME)
    (FORMAT.DEF (PARSE.ATOM (GET.ARG.SAV NAME))
           '(Property Name)
           (SAVE.DUMPOUT FONT LISP DUMP.CHARS (GET.ARG.SAV NAME])

(RBRACKET#TOPROG
  [LAMBDA NIL                                                    (* mjs "10-Apr-85 09:50")
    (IM.DUMP.CHARS "}")
    (TRIVIAL.ARG])

(REF#TOPROG
  [LAMBDA NIL                                                    (* mjs " 4-Aug-86 11:40")
    (if (NOT IM.REF.FLG)
        then (SAVE.ARG)
             (IM.DUMP.CHARS (SELECTQ TO.NAME
                                 (PAGEREF "page X.XX")
                                 (SECTIONREF "section X.XX")
                                 (FIGUREREF "figure X.X")
                                 (SHOULDNT)))
      else (PROG ((SAV (SAVE.ARG))
                  (DEF.REFS NIL)
                  (PRIMARY.REFS NIL)
                  (SECONDARY.REFS NIL)
                  (MAX.REF NIL)
                  REF.STRING TYPE ARGS TEMP ARG.ATOM REFS)
                 (SETQ TEMP (PARSE.INDEX.SPEC SAV NIL))
                 (if (OR (NULL TEMP)
                         (NULL (CAR TEMP)))
                     then (IM.WARNING "null index --- ignored")
                          (RETURN))
                 (SETQ ARGS (CAR TEMP))
                 (SETQ TYPE (if (EQ TO.NAME 'FIGUREREF)
                                then                             (* for FIGUREREF, ignore specified type --- use TAG)
                                     'TAG
                                elseif (U-CASE (CDR TEMP))
                                else 'TERM))
                 [SETQ ARG.ATOM (U-CASE (MKATOM (LIST.TO.STRING ARGS]

          (* * only look at refs that have the correct type, AND that are not subentries or subsub entries)

                 (SETQ REFS (for X in (GETHASH ARG.ATOM IMPTR.HASH)
                               when [AND (EQUAL (if (U-CASE (fetch (IM.INDEX.DATA TYPE) of X))
                                                    else 'TERM)
                                                TYPE)
                                         (NULL (LISTGET (fetch (IM.INDEX.DATA PROPLIST) of X)
                                                        'SUBNAME))
                                         (NULL (LISTGET (fetch (IM.INDEX.DATA PROPLIST) of X)
                                                        'SUBTEXT]
                               collect X))
                 (if (NULL REFS)
                     then (IM.WARNING " no refs for resolving {" TO.NAME " " TYPE " " ARG.ATOM "} -- dummy used")
                          (IM.DUMP.CHARS (SELECTQ TO.NAME
                                              (PAGEREF "page X.XX")
                                              (SECTIONREF "section X.XX")
                                              (FIGUREREF "figure X.X")
                                              (SHOULDNT)))
                          (RETURN))

          (* * REFS is list list of refs to index name ARG.ATOM of type TYPE, with elements of form%:
          (type text info section file fileptr))

                 [for X in REFS do (if (OR (AND (EQ TO.NAME 'FIGUREREF)
                                                (MEMB '*FIGURE* (fetch (IM.INDEX.DATA INFO) of X)))
                                           (MEMB '*PRIMARY* (fetch (IM.INDEX.DATA INFO) of X)))
                                       then (SETQ PRIMARY.REFS (CONS X PRIMARY.REFS))
                                     elseif (MEMB '*DEF* (fetch (IM.INDEX.DATA INFO) of X))
                                       then (SETQ DEF.REFS (CONS X DEF.REFS))
                                     else (SETQ SECONDARY.REFS (CONS X SECONDARY.REFS]
                 (SETQ MAX.REF (if PRIMARY.REFS
                                   elseif DEF.REFS
                                   else SECONDARY.REFS))
                 (if (CDR MAX.REF)
                     then (IM.WARNING "multiple " (if PRIMARY.REFS
                                                      then "primary"
                                                    elseif DEF.REFS
                                                      then "def"
                                                    else "secondary")
                                      " refs for resolving {" TO.NAME " " TYPE " " ARG.ATOM "} - first used"))
                 (SETQ MAX.REF (CAR MAX.REF))
                 (SETQ REF.STRING (SELECTQ TO.NAME
```

```
                                    (PAGEREF [PROG ((CHAP.PAGE.LST (REF.TO.PAGE MAX.REF)))
                                                   (RETURN (if (EQ 0 (CAR CHAP.PAGE.LST))
                                                               then (CONCAT "page " (CADR CHAP.PAGE.LST))
                                                             else (CONCAT "page " (CAR CHAP.PAGE.LST)
                                                                          "."
                                                                          (CADR CHAP.PAGE.LST])
                                    (SECTIONREF (PROG ((SEC.LIST (REVERSE (fetch (IM.INDEX.DATA SUBSEC)
                                                                                 of X)))
                                                       SEC.STRING)
                                                      (SETQ SEC.STRING (if (CDR SEC.LIST)
                                                                           then "section "
                                                                         elseif (NUMBERP (CAR SEC.LIST))
                                                                           then "chapter "
                                                                         else "appendix "))
                                                      (if (EQ 0 (CAR SEC.LIST))
                                                          then (SETQ SEC.LIST (CDR SEC.LIST)))
                                                      [for X on SEC.LIST
                                                         do (SETQ SEC.STRING (CONCAT SEC.STRING (CAR X)
                                                                                     (if (CDR X)
                                                                                         then "."
                                                                                       else ""]
                                                      (RETURN SEC.STRING)))
                                    (FIGUREREF [PROG [[CHAP.NUM (CAR (LAST (fetch (IM.INDEX.DATA SUBSEC)
                                                                                  of X]
                                                      (FIG.NUM (CADR (MEMB '*FIGURE* (fetch (IM.INDEX.DATA
                                                                                            INFO)
                                                                                     of X]
                                                     (RETURN (if (EQ 0 CHAP.NUM)
                                                                 then (CONCAT "figure " FIG.NUM)
                                                               else (CONCAT "figure " CHAP.NUM "." FIG.NUM])
                                    (SHOULDNT)))
          (IM.DUMP.CHARS REF.STRING])
```

(RM#TOPROG

```
  [LAMBDA NIL                                                    (* mjs " 4-MAY-83 10:23")
    (DUMPOUT FONT NIL DUMP.ARG])
```

(SUB#TOPROG

```
  [LAMBDA NIL                                                    (* mjs "14-Dec-83 10:44")
    (DUMPOUT START.SUB DUMP.ARG END.SUB])
```

(SUBSEC#TOPROG

```
  [LAMBDA NIL                                                    (* mjs " 3-Oct-85 15:00")
    (SAVE.ARGS TITLE)
    (PROG ((SUBSEC.COUNT.LIST (CONS (SETQ SUBSEC.LAST.SUB (ADD1 SUBSEC.LAST.SUB))
                                    SUBSEC.COUNT.LIST))
           (SUBSEC.LAST.SUB 0)
           PRINTING.TITLE SEC.STRING SEC.LIST CHAP.NUM)
          (DECLARE (SPECVARS SUBSEC.COUNT.LIST SUBSEC.LAST.SUB))

          (* SUBSEC.COUNT.LIST is a reverse list of the subsec numbers and chapter num, so if this is subsec 3.5.7,
          SUBSEC.COUNT.LIST = (7 5 3))
                                                                 (* set SUBSEC.SKIP.STRING to skip before header
                                                                 (<<<DELETED IN IM.TEDIT>>>))
                                                                 (* set PRINTING.TITLE to subsec title <u-case if 1st-level
                                                                 subsec)
          (SETQ PRINTING.TITLE (GET.ARG.SAV TITLE))
          (SETQ SEC.STRING "")
          (SETQ SEC.LIST (REVERSE SUBSEC.COUNT.LIST))
          [SETQ CHAP.NUM (CAR (MKLIST (CAR SEC.LIST]
          [SETQ SEC.LIST (if (NULL CHAP.NUM)
                             then (CDR SEC.LIST)
                           else (CONS CHAP.NUM (CDR SEC.LIST]
          [for X on SEC.LIST do (SETQ SEC.STRING (CONCAT SEC.STRING (CAR X)
                                                         (if (CDR X)
                                                             then "."
                                                           else ""]
          [if (EQ 2 (LENGTH SUBSEC.COUNT.LIST))
              then                                               (* major heading)
                   (DUMP.HEADERS.FOOTERS (U-CASE (PARSE.STRING PRINTING.TITLE))
                          NIL)
                   (DUMP.HRULE 2 55
                          '(QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 LINELEADING 0 PARALEADING 0 POSTPARALEADING 0
                                 HEADINGKEEP ON))
                   (DUMPOUT FONT IM.SUBSEC.ONE.TITLE.FONT PARALOOKS
                          `(QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 LINELEADING 0 PARALEADING 0 POSTPARALEADING 0
                                 TABS %, IM.SUBSEC.TITLE.TABS HEADINGKEEP ON)
                          DUMP.CHARS SEC.STRING DUMP.CHARS "  " TAB DUMP.CHARS PRINTING.TITLE CR CR)
                   (DUMP.HRULE 1 NIL
                          '(QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 LINELEADING 0 PARALEADING 0 POSTPARALEADING 0
                                 HEADINGKEEP ON))
            elseif (EQ 3 (LENGTH SUBSEC.COUNT.LIST))
              then                                               (* important heading)
                   (DUMPOUT FONT IM.SUBSEC.TWO.TITLE.FONT PARALOOKS
```

```
                        `(QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 LINELEADING 0 PARALEADING 35 POSTPARALEADING
                              0 TABS %, IM.SUBSEC.TITLE.TABS HEADINGKEEP ON)
                        DUMP.CHARS SEC.STRING DUMP.CHARS "  " TAB DUMP.CHARS PRINTING.TITLE CR CR)
                  (DUMP.HRULE 1 NIL
                        '(QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 LINELEADING 0 PARALEADING 0 POSTPARALEADING 0
                              HEADINGKEEP ON))
            else                                                    (* plain heading for 3rd or greater level headings)
                 (DUMPOUT FONT IM.SUBSEC.THREE.TITLE.FONT PARALOOKS
                        `(QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 LINELEADING 0 PARALEADING 35 POSTPARALEADING 0
                              TABS %, IM.SUBSEC.TITLE.TABS HEADINGKEEP ON)
                        DUMP.CHARS SEC.STRING DUMP.CHARS "  " TAB DUMP.CHARS PRINTING.TITLE CR CR)
                 (DUMP.HRULE 1 NIL
                        '(QUAD LEFT 1STLEFTMARGIN 0 LEFTMARGIN 0 LINELEADING 0 PARALEADING 0 POSTPARALEADING 0
                              HEADINGKEEP ON]
           (SEND.INFO [U-CASE (MKATOM (PARSE.STRING (GET.ARG.SAV TITLE]
                  'SUBSEC
                  (GET.ARG.SAV TITLE))
           (if (EQ TO.ARG.NAME 'TEXT)
               then (DUMP.ARG)
                    (DUMPOUT CR CR])
```

(SUPER#TOPROG

```
  [LAMBDA NIL                                                       (* mjs "14-Dec-83 10:44")
    (DUMPOUT START.SUPER DUMP.ARG END.SUPER])
```

(TABLE#TOPROG

```
  [LAMBDA NIL                                                       (* mjs "25-Jul-85 10:33")
    (IM.ERROR "Table --- de-implemented")
    [IM.HOLD.FOOTNOTES (bind ARG.NAME while (SETQ ARG.NAME (GET.ARG))
                          do (if (MEMB ARG.NAME '(VSKIP HSKIP COLUMN MULTIPAGE UNDERLINE))
                                 then (FLUSH.ARG)
                               else (DUMPOUT DUMP.ARG CR CR]

           (* * old version%: (IM.HOLD.FOOTNOTES (PROG ((TotalWidth
           (ANC.WIDTH)) (CurrentIndent (ANC.INDENT)) (FORMAT.PARSED NIL)
           (vskip 10) (hskip 15) (ColumnWidthList NIL) (UNDERLINE.FLG NIL)
           (MULTIPAGE.FLG NIL) (BEGIN.FLG T) TableWidth numcol freecol ARG.NAME col COLUMN.INDENT.WIDTH.LIST
           COLUMN.WIDTH COLUMN.INDENT COLUMN.PARALOOKS BEGIN.ROW.FLG) loop
           (SELECTQ (SETQ ARG.NAME (GET.ARG)) (VSKIP (if FORMAT.PARSED then
           (FLUSH.ARG) else (SETQ vskip (PARSE.NUMBER.OR.PERCENTAGE
           (SAVE.ARG) 10 vskip)))) (HSKIP (if FORMAT.PARSED then (FLUSH.ARG) else
           (SETQ hskip (PARSE.NUMBER.OR.PERCENTAGE (SAVE.ARG) TotalWidth hskip))))
           (COLUMN (if FORMAT.PARSED then (FLUSH.ARG) else (SETQ ColumnWidthList
           (CONS (PARSE.NUMBER.OR.PERCENTAGE (SAVE.ARG) TotalWidth NIL) ColumnWidthList))))
           (MULTIPAGE (if (NOT FORMAT.PARSED) then (SETQ MULTIPAGE.FLG T))
           (FLUSH.ARG)) (UNDERLINE (SETQ UNDERLINE.FLG T) (FLUSH.ARG))
           (NIL (RETURN)) (PROGN (if (NOT FORMAT.PARSED) then (* default format spec = 3 columns)
           (SETQ ColumnWidthList (if (NULL ColumnWidthList) then (LIST NIL NIL NIL) else
           (DREVERSE ColumnWidthList))) (SETQ numcol (LENGTH ColumnWidthList))
           (SETQ TableWidth (IPLUS (ITIMES hskip (SUB1 numcol)) (for X in ColumnWidthList when
           (NUMBERP X) sum X))) (SETQ freecol (for X in ColumnWidthList count
           (NULL X))) (if (GREATERP TableWidth TotalWidth) then (IM.WARNING "Table Spec too big --- fudging available space
           (may cause overlapping") (SETQ TotalWidth (FIX (FTIMES TableWidth 1.1))))
           (if (GREATERP freecol 0) then (* divide remaining space among unspecified columns)
           (for X on ColumnWidthList when (NULL (CAR X)) do (RPLACA X
           (IQUOTIENT (IDIFFERENCE TotalWidth TableWidth) freecol)))
           (SETQ TableWidth TotalWidth)) (SETQ COLUMN.INDENT.WIDTH.LIST
           (for X in ColumnWidthList bind (I _ 0) collect (PROG1 (CONS I X)
           (SETQ I (IPLUS I X hskip))))) (SETQ col NIL) (SETQ FORMAT.PARSED T))
           (if (AND col (EQ ARG.NAME (QUOTE FIRST))) then (* if you have a "first" column item, and you are still in a line, close the
           line) (SETQ col NIL)) (if (SETQ BEGIN.ROW.FLG (NULL col)) then
           (SETQ col COLUMN.INDENT.WIDTH.LIST)) (SETQ COLUMN.INDENT
           (CAR (CAR col))) (SETQ COLUMN.WIDTH (CDR (CAR col))) (* specify PARALOOKS of left-justified, right-margin)
           (SETQ COLUMN.PARALOOKS (CONS (QUOTE RIGHTMARGIN)
           (CONS (IPLUS COLUMN.INDENT COLUMN.WIDTH) (QUOTE
           (QUAD LEFT))))) (if BEGIN.FLG then (* for very first para of table only, use default PARALEADING)
           (SETQ BEGIN.FLG NIL) elseif BEGIN.ROW.FLG then (* before a FIRST column, use PARLEADING of vskip)
           (SETQ COLUMN.PARALOOKS (CONS (QUOTE PARALEADING)
           (CONS vskip COLUMN.PARALOOKS))) else (* before a NEXT column, use PARALEADING of 0)
           (SETQ COLUMN.PARALOOKS (CONS (QUOTE PARALEADING)
           (CONS 0 COLUMN.PARALOOKS)))) (DUMPOUT WIDTH (CDR
           (CAR col)) PARALOOKS COLUMN.PARALOOKS START.PARA DUMP.ARG CR CR)
           (SETQ col (CDR col)) (* currently, don't use underline) (SETQ UNDERLINE.FLG NIL)))
           (GO loop))))

    ])
```

(TAG#TOPROG

```
  [LAMBDA NIL                                                       (* mjs "27-JUN-83 15:13")
    (PROG ((SAV (SAVE.ARG)))
          (SEND.INFO (U-CASE (PARSE.ATOM SAV))
                 'TAG NIL)
          (SAVE.INFILE.NOTE 'IM.TAG (PARSE.ATOM SAV])
```

```
(TERM#TOPROG
  [LAMBDA NIL                                                    (* mjs "10-Apr-85 09:49")
    (PROG ((SAV (SAVE.ARG)))
          (IM.DUMP.CHARS SAV)
          [SETQ ARG.ATOM (MKATOM (LIST.TO.STRING (PARSE.LIST SAV]
          (SEND.INFO (U-CASE ARG.ATOM)
                 'TERM ARG.ATOM NIL])

(VAR#TOPROG
  [LAMBDA NIL                                                    (* mjs "13-SEP-83 17:15")
    (PROG ((SAV (SAVE.ARG)))
          (FORMAT.LISPWORD SAV)
          (SEND.IMPLICIT (PARSE.ATOM SAV)
                 '(Variable)
                 SAV)
          (if [AND IM.CHECK.DEFS (NOT (BOUNDP (PARSE.ATOM SAV]
              then (SAVE.INFILE.NOTE 'UNBOUND.VAR (PARSE.ATOM SAV])

(VARDEF#TOPROG
  [LAMBDA NIL                                                    (* mjs "10-Apr-85 11:17")
    (SAVE.ARGS NAME)
    (PROG [(NAME (PARSE.ATOM (GET.ARG.SAV NAME]
          (if IM.CHECK.DEFS
              then (if (NOT (BOUNDP NAME))
                       then (PRINT.NOTE (CONCAT NAME " is unbound in Interlisp-D"))
                            (IM.WARNING NAME " is defined as a variable, but is unbound in Interlisp-D")
                            (SAVE.INFILE.NOTE 'UNBOUND.VAR NAME)))
          (FORMAT.DEF NAME '(Variable)
                 (SAVE.DUMPOUT FONT LISP DUMP.CHARS (GET.ARG.SAV NAME])

)

(RPAQQ TO.NAME.LIST
       (ANONARG ARG ATOM BIGLISPCODE BRACKET BREAKCOM BULLET CHAPTER COMMENT CRSYMBOL DEF EDITCOM ELLIPSIS
              EMDASH ENDASH FIGURE FIGUREREF FILECOM FN FNDEF FOOT GE INCLUDE INDEX INDEXX IT LABELEDLIST
              LBRACKET LE LISP LISPCODE MAC MACDEF NE NOTE NUMBEREDLIST PACOM PAGEREF PI PLUSMINUS PROP PROPDEF
              RBRACKET RM SECTIONREF SP SUB SUBSEC SUPER TABLE TAG TERM UNNUMBEREDLIST VAR VARDEF))

(RPAQQ TO.SYNONYM.LIST ((CR CRSYMBOL)
                        (EMPHASIZE IT)
                        (FOOTNOTE FOOT)
                        (ITALICS IT)
                        (LITATOM ATOM)
                        (UNLABELEDLIST UNNUMBEREDLIST)))

(RPAQQ TO.NAME.LIST
       (ANONARG ARG ATOM BIGLISPCODE BRACKET BREAKCOM BULLET CHAPTER COMMENT CRSYMBOL DEF EDITCOM ELLIPSIS
              EMDASH ENDASH FIGURE FIGUREREF FILECOM FN FNDEF FOOT GE INCLUDE INDEX INDEXX IT LABELEDLIST
              LBRACKET LE LISP LISPCODE MAC MACDEF NE NOTE NUMBEREDLIST PACOM PAGEREF PI PLUSMINUS PROP PROPDEF
              RBRACKET RM SECTIONREF SP SUB SUBSEC SUPER TABLE TAG TERM UNNUMBEREDLIST VAR VARDEF))

(PUTPROPS ANONARG TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS ARG TO.PROG ARG#TOPROG)

(PUTPROPS ATOM TO.PROG LISPWORD#TOPROG)

(PUTPROPS BIGLISPCODE TO.PROG BIGLISPCODE#TOPROG)

(PUTPROPS BRACKET TO.PROG BRACKET#TOPROG)

(PUTPROPS BREAKCOM TO.PROG LISPWORD#TOPROG)

(PUTPROPS BULLET TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS CHAPTER TO.PROG CHAPTER#TOPROG)

(PUTPROPS COMMENT TO.PROG COMMENT#TOPROG)

(PUTPROPS CRSYMBOL TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS DEF TO.PROG DEF#TOPROG)

(PUTPROPS EDITCOM TO.PROG LISPWORD#TOPROG)

(PUTPROPS ELLIPSIS TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS EMDASH TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS ENDASH TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS FIGURE TO.PROG FIGURE#TOPROG)

(PUTPROPS FIGUREREF TO.PROG REF#TOPROG)
```

```
(PUTPROPS FILECOM TO.PROG LISPWORD#TOPROG)

(PUTPROPS FN TO.PROG FN#TOPROG)

(PUTPROPS FNDEF TO.PROG FNDEF#TOPROG)

(PUTPROPS FOOT TO.PROG FOOT#TOPROG)

(PUTPROPS GE TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS INCLUDE TO.PROG INCLUDE#TOPROG)

(PUTPROPS INDEX TO.PROG INDEX#TOPROG)

(PUTPROPS INDEXX TO.PROG INDEXX#TOPROG)

(PUTPROPS IT TO.PROG IT#TOPROG)

(PUTPROPS LABELEDLIST TO.PROG LIST#TOPROG)

(PUTPROPS LBRACKET TO.PROG LBRACKET#TOPROG)

(PUTPROPS LE TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS LISP TO.PROG LISP#TOPROG)

(PUTPROPS LISPCODE TO.PROG LISPCODE#TOPROG)

(PUTPROPS MAC TO.PROG LISPWORD#TOPROG)

(PUTPROPS MACDEF TO.PROG MACDEF#TOPROG)

(PUTPROPS NE TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS NOTE TO.PROG NOTE#TOPROG)

(PUTPROPS NUMBEREDLIST TO.PROG LIST#TOPROG)

(PUTPROPS PACOM TO.PROG LISPWORD#TOPROG)

(PUTPROPS PAGEREF TO.PROG REF#TOPROG)

(PUTPROPS PI TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS PLUSMINUS TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS PROP TO.PROG LISPWORD#TOPROG)

(PUTPROPS PROPDEF TO.PROG PROPDEF#TOPROG)

(PUTPROPS RBRACKET TO.PROG RBRACKET#TOPROG)

(PUTPROPS RM TO.PROG RM#TOPROG)

(PUTPROPS SECTIONREF TO.PROG REF#TOPROG)

(PUTPROPS SP TO.PROG PRINT.SPECIAL.CHARS#TOPROG)

(PUTPROPS SUB TO.PROG SUB#TOPROG)

(PUTPROPS SUBSEC TO.PROG SUBSEC#TOPROG)

(PUTPROPS SUPER TO.PROG SUPER#TOPROG)

(PUTPROPS TABLE TO.PROG TABLE#TOPROG)

(PUTPROPS TAG TO.PROG TAG#TOPROG)

(PUTPROPS TERM TO.PROG TERM#TOPROG)

(PUTPROPS UNNUMBEREDLIST TO.PROG LIST#TOPROG)

(PUTPROPS VAR TO.PROG VAR#TOPROG)

(PUTPROPS VARDEF TO.PROG VARDEF#TOPROG)

(PUTPROPS CHAPTER TO.ARGS ((TITLE NUMBER)
                           TEXT))

(PUTPROPS DEF TO.ARGS ((TYPE NAME PRINTNAME ARGS PARENS NOPARENS)
                       TEXT))

(PUTPROPS FIGURE TO.ARGS ((TAG)
                          (TEXT)
                          (CAPTION)))
```

```
(PUTPROPS FNDEF TO.ARGS (NAME (ARGS)
                              (TYPE)
                             TEXT))

(PUTPROPS INDEXX TO.ARGS ((TYPE NAME INFO TEXT SUBNAME SUBTYPE SUBTEXT SUBSUBNAME SUBSUBTYPE SUBSUBTEXT)))

(PUTPROPS LABELEDLIST TO.ARGS ((LNAME NAME ITEM INDENT MAX UNINDENTED)))

(PUTPROPS MACDEF TO.ARGS (NAME (ARGS)
                               (TYPE)
                              TEXT))

(PUTPROPS NUMBEREDLIST TO.ARGS ((ITEM)))

(PUTPROPS PROPDEF TO.ARGS (NAME TEXT))

(PUTPROPS SUBSEC TO.ARGS (TITLE TEXT))

(PUTPROPS TABLE TO.ARGS ((FIRST NEXT COLUMN UNDERLINE MULTIPAGE HSKIP VSKIP)))

(PUTPROPS UNNUMBEREDLIST TO.ARGS ((ITEM)))

(PUTPROPS VARDEF TO.ARGS (NAME TEXT))

(PUTPROPS FNDEF TO.ARG.SYNONYMS (FNNAME NAME FNARGS ARGS FNTYPE TYPE))

(PUTPROPS LABELEDLIST TO.ARG.SYNONYMS (LABEL NAME TEXT ITEM UNINDENT UNINDENTED UNLABELED UNINDENTED UNLABEL
                                       UNINDENTED))

(PUTPROPS NUMBEREDLIST TO.ARG.SYNONYMS (TEXT ITEM))

(PUTPROPS TABLE TO.ARG.SYNONYMS (COL COLUMN MULTI MULTIPAGE))

(PUTPROPS UNNUMBEREDLIST TO.ARG.SYNONYMS (TEXT ITEM))

(PUTPROPS ANONARG TO.TYPE SIMPLE)

(PUTPROPS ARG TO.TYPE SIMPLE)

(PUTPROPS ATOM TO.TYPE SIMPLE)

(PUTPROPS BIGLISPCODE TO.TYPE NIL)

(PUTPROPS BRACKET TO.TYPE SIMPLE)

(PUTPROPS BREAKCOM TO.TYPE SIMPLE)

(PUTPROPS BULLET TO.TYPE SIMPLE)

(PUTPROPS CHAPTER TO.TYPE NIL)

(PUTPROPS COMMENT TO.TYPE SIMPLE)

(PUTPROPS CRSYMBOL TO.TYPE SIMPLE)

(PUTPROPS EDITCOM TO.TYPE SIMPLE)

(PUTPROPS ELLIPSIS TO.TYPE SIMPLE)

(PUTPROPS EMDASH TO.TYPE SIMPLE)

(PUTPROPS ENDASH TO.TYPE SIMPLE)

(PUTPROPS FIGURE TO.TYPE NIL)

(PUTPROPS FIGUREREF TO.TYPE SIMPLE)

(PUTPROPS FILECOM TO.TYPE SIMPLE)

(PUTPROPS FN TO.TYPE SIMPLE)

(PUTPROPS FNDEF TO.TYPE NIL)

(PUTPROPS FOOT TO.TYPE SIMPLE)

(PUTPROPS GE TO.TYPE SIMPLE)

(PUTPROPS INCLUDE TO.TYPE SIMPLE)

(PUTPROPS INDEX TO.TYPE SIMPLE)

(PUTPROPS INDEXX TO.TYPE SIMPLE)

(PUTPROPS IT TO.TYPE SIMPLE)

(PUTPROPS LABELEDLIST TO.TYPE NIL)
```

(PUTPROPS **LBRACKET TO.TYPE** SIMPLE)

(PUTPROPS **LE TO.TYPE** SIMPLE)

(PUTPROPS **LISP TO.TYPE** SIMPLE)

(PUTPROPS **LISPCODE TO.TYPE** NIL)

(PUTPROPS **MAC TO.TYPE** SIMPLE)

(PUTPROPS **MACDEF TO.TYPE** NIL)

(PUTPROPS **NE TO.TYPE** SIMPLE)

(PUTPROPS **NOTE TO.TYPE** SIMPLE)

(PUTPROPS **NUMBEREDLIST TO.TYPE** NIL)

(PUTPROPS **PACOM TO.TYPE** SIMPLE)

(PUTPROPS **PAGEREF TO.TYPE** SIMPLE)

(PUTPROPS **PI TO.TYPE** SIMPLE)

(PUTPROPS **PLUSMINUS TO.TYPE** SIMPLE)

(PUTPROPS **PROP TO.TYPE** SIMPLE)

(PUTPROPS **PROPDEF TO.TYPE** NIL)

(PUTPROPS **RBRACKET TO.TYPE** SIMPLE)

(PUTPROPS **RM TO.TYPE** SIMPLE)

(PUTPROPS **SECTIONREF TO.TYPE** SIMPLE)

(PUTPROPS **SP TO.TYPE** SIMPLE)

(PUTPROPS **SUB TO.TYPE** SIMPLE)

(PUTPROPS **SUBSEC TO.TYPE** NIL)

(PUTPROPS **SUPER TO.TYPE** SIMPLE)

(PUTPROPS **TABLE TO.TYPE** NIL)

(PUTPROPS **TAG TO.TYPE** SIMPLE)

(PUTPROPS **TERM TO.TYPE** SIMPLE)

(PUTPROPS **UNNUMBEREDLIST TO.TYPE** NIL)

(PUTPROPS **VAR TO.TYPE** SIMPLE)

(PUTPROPS **VARDEF TO.TYPE** NIL)

(PUTPROPS **ANONARG TO.ARG.TYPE** CHARS)

(PUTPROPS **ARG TO.ARG.TYPE** SIMPLE)

(PUTPROPS **ATOM TO.ARG.TYPE** SIMPLE)

(PUTPROPS **BIGLISPCODE TO.ARG.TYPE** SIMPLE)

(PUTPROPS **BRACKET TO.ARG.TYPE** SIMPLE)

(PUTPROPS **BREAKCOM TO.ARG.TYPE** SIMPLE)

(PUTPROPS **BULLET TO.ARG.TYPE** CHARS)

(PUTPROPS **CHAPTER TO.ARG.TYPE** (TITLE SIMPLE NUMBER CHARS))

(PUTPROPS **COMMENT TO.ARG.TYPE** NIL)

(PUTPROPS **CRSYMBOL TO.ARG.TYPE** CHARS)

(PUTPROPS **DEF TO.ARG.TYPE** (TYPE CHARS NAME SIMPLE PRINTNAME SIMPLE PARENS CHARS NOPARENS CHARS))

(PUTPROPS **EDITCOM TO.ARG.TYPE** SIMPLE)

(PUTPROPS **ELLIPSIS TO.ARG.TYPE** CHARS)

(PUTPROPS **EMDASH TO.ARG.TYPE** CHARS)

(PUTPROPS **ENDASH TO.ARG.TYPE** CHARS)

(PUTPROPS **FIGURE TO.ARG.TYPE** (TAG CHARS))

```
(PUTPROPS FIGUREREF TO.ARG.TYPE CHARS)

(PUTPROPS FILECOM TO.ARG.TYPE SIMPLE)

(PUTPROPS FN TO.ARG.TYPE SIMPLE)

(PUTPROPS FNDEF TO.ARG.TYPE (NAME SIMPLE ARGS SIMPLE TYPE CHARS))

(PUTPROPS FOOT TO.ARG.TYPE SIMPLE)

(PUTPROPS GE TO.ARG.TYPE CHARS)

(PUTPROPS INCLUDE TO.ARG.TYPE CHARS)

(PUTPROPS INDEX TO.ARG.TYPE CHARS)

(PUTPROPS INDEXX TO.ARG.TYPE (TYPE CHARS NAME CHARS INFO CHARS TEXT SIMPLE SUBNAME CHARS SUBTYPE CHARS SUBTEXT
                                  SIMPLE SUBSUBNAME CHARS SUBSUBTYPE CHARS SUBSUBTEXT SIMPLE))

(PUTPROPS IT TO.ARG.TYPE SIMPLE)

(PUTPROPS LABELEDLIST TO.ARG.TYPE (LNAME SIMPLE NAME SIMPLE INDENT CHARS MAX CHARS))

(PUTPROPS LBRACKET TO.ARG.TYPE CHARS)

(PUTPROPS LE TO.ARG.TYPE CHARS)

(PUTPROPS LISP TO.ARG.TYPE SIMPLE)

(PUTPROPS LISPCODE TO.ARG.TYPE SIMPLE)

(PUTPROPS MAC TO.ARG.TYPE SIMPLE)

(PUTPROPS MACDEF TO.ARG.TYPE (NAME SIMPLE ARGS SIMPLE TYPE CHARS))

(PUTPROPS NE TO.ARG.TYPE CHARS)

(PUTPROPS NOTE TO.ARG.TYPE NIL)

(PUTPROPS NUMBEREDLIST TO.ARG.TYPE NIL)

(PUTPROPS PACOM TO.ARG.TYPE SIMPLE)

(PUTPROPS PAGEREF TO.ARG.TYPE CHARS)

(PUTPROPS PI TO.ARG.TYPE CHARS)

(PUTPROPS PLUSMINUS TO.ARG.TYPE CHARS)

(PUTPROPS PROP TO.ARG.TYPE SIMPLE)

(PUTPROPS PROPDEF TO.ARG.TYPE (NAME SIMPLE))

(PUTPROPS RBRACKET TO.ARG.TYPE CHARS)

(PUTPROPS RM TO.ARG.TYPE SIMPLE)

(PUTPROPS SECTIONREF TO.ARG.TYPE CHARS)

(PUTPROPS SP TO.ARG.TYPE CHARS)

(PUTPROPS SUB TO.ARG.TYPE SIMPLE)

(PUTPROPS SUBSEC TO.ARG.TYPE (TITLE SIMPLE))

(PUTPROPS SUPER TO.ARG.TYPE SIMPLE)

(PUTPROPS TABLE TO.ARG.TYPE (COLUMN CHARS UNDERLINE CHARS MULTIPAGE CHARS HSKIP CHARS VSKIP CHARS))

(PUTPROPS TAG TO.ARG.TYPE CHARS)

(PUTPROPS TERM TO.ARG.TYPE SIMPLE)

(PUTPROPS UNNUMBEREDLIST TO.ARG.TYPE NIL)

(PUTPROPS VAR TO.ARG.TYPE SIMPLE)

(PUTPROPS VARDEF TO.ARG.TYPE (NAME SIMPLE))

(RPAQ? IM.TEDIT.FONT.DEFS
       '(NIL (FAMILY MODERN FACE MRR SIZE 10)
             FOOTNOTE
             (FAMILY MODERN FACE MRR SIZE 10)
             NOTE
             (FAMILY MODERN FACE MIR SIZE 8)
             BOLD
```

```
            (FAMILY MODERN FACE BRR SIZE 10)
            ITALIC
            (FAMILY MODERN FACE MIR SIZE 10)
            LISP
            (FAMILY MODERN FACE BRR SIZE 10)
            ARG
            (FAMILY MODERN FACE MIR SIZE 10)))

(RPAQ? IM.CHAPTER.TITLE.FONT '(FAMILY MODERN FACE BRR SIZE 18))

(RPAQ? IM.SUBSEC.ONE.TITLE.FONT '(FAMILY MODERN SIZE 14 FACE BRR))

(RPAQ? IM.SUBSEC.TWO.TITLE.FONT '(FAMILY MODERN SIZE 12 FACE BRR))

(RPAQ? IM.SUBSEC.THREE.TITLE.FONT '(FAMILY MODERN SIZE 10 FACE BRR))

(RPAQ? IM.TEXT.FONT '(FAMILY MODERN FACE MRR SIZE 10))

(RPAQ? IM.HEADER.FOOTER.FONT '(FAMILY MODERN FACE MRR SIZE 8))

(RPAQ? IM.XEROX.LOGO.FONT '(FAMILY MODERN FACE BRR SIZE 30))

;; the following variables specify all of the lengths used for possitioning IM text, headers, etc.  on the page.  All of these are measured with respect to the
;; page *margins* <the region on the page defined by the Tedit margin parameters> or with respect to the page *edges* <the edges of the physical
;; page>.

;; Note: The formatting and printing does not always position the image on the page exactly as specified.  It will probably be necessary to adjust any
;; variables based on the page edges until they come out correctly on your printer.

;; indentation of 1st line of definitian header, measured in points from left page margin.  Also used for indentation of hrule under defn header.

(RPAQ? IM.DEF.TITLE.1STLEFTMARGIN 75)

;; indentation of 2nd and other overflow lines of definition header, measured in points from left page margin.

(RPAQ? IM.DEF.TITLE.LEFTMARGIN 204)

;; indentation of vertical rule to the left of definition text, measured in points from left page margin.  This is a never-used, never-debugged feature.

(RPAQ? IM.VRULE.X 194)

;; y-pos of top-left corner of top text line, measured in points from bottom page edge.

(RPAQ? IM.TEXT.TOPMARGIN 738)

;; y-pos of bottom-left corner of bottom text line, measured in points from bottom page edge.

(RPAQ? IM.TEXT.BOTTOMMARGIN 54)

;; x-pos of left edge of text, measured in points from the left page margin.

(RPAQ? IM.TEXT.LEFTMARGIN 204)

;; x-pos of right edge of text, measured in points from the left page margin.

(RPAQ? IM.TEXT.RIGHTMARGIN 504)

;; X-pos and Y-pos of the lower-left corner of the '[This page intentionally left blank]' message printed on blank pages, measured in points from the left
;; and bottom page edges.

(RPAQ? IM.BLANKPAGE.SPECIALX 258)

(RPAQ? IM.BLANKPAGE.SPECIALY 400)

;; In the table of contents, indentation of first and second-level subsection headers, measured in points from the left page margin.

(RPAQ? IM.TOC.SUBSEC.ONE.LEFTMARGIN 120)

(RPAQ? IM.TOC.SUBSEC.TWO.LEFTMARGIN 216)

;; in the index, the indentation of the first line and remaining lines of a top-level entry, of a subentry, and of a subsubentry, measured in points from the
;; left page margin <for the left column>.

(RPAQ? IM.INDEX.1STLEFTMARGIN 0)

(RPAQ? IM.INDEX.LEFTMARGIN 75)

(RPAQ? IM.INDEX.SUB.1STLEFTMARGIN 25)

(RPAQ? IM.INDEX.SUB.LEFTMARGIN 75)

(RPAQ? IM.INDEX.SUBSUB.1STLEFTMARGIN 50)
```

```
(RPAQ? IM.INDEX.SUBSUB.LEFTMARGIN 75)

;; on the title page, the y-pos of the lower-left corner of the first line in the title <and of the XEROX logo>, measured in points from the bottom page
;; margin.  The X-pos is always 0 for the XEROX logo, and the normal text indentation for the title.

(RPAQ? IM.TITLEPAGE.TITLE.Y 258)

;; on the title page, the y-pos of the lower-left corner of the first line in the document number, measured in points from the bottom page margin.  The
;; Y-pos is always the normal text indentation.

(RPAQ? IM.TITLEPAGE.DOCNUMBER.Y 45)

;; Tedit tab setting used for subsection heading text.  '(40 . LEFT)' determines the indentation of the title after the subsec number, measured in points
;; from the left page margin.  '18' is the tab used if the subsec number is wider than 40 pts.

(RPAQ? IM.SUBSEC.TITLE.TABS '(18 (40 . LEFT)))

;; Tedit tab setting used for chapter titles, headers, and footers to right-justify text.  '(504 . RIGHT)' specifies a right tab at the right-hand edge of the text,
;; measured in points from the left page margin.

(RPAQ? IM.RIGHT.MARGIN.TABS '(0 (504 . RIGHT)))

;; Tedit tab setting used for labeled lists, numbered lists, bullet-ed lists.  '(186 . RIGHT)' right-justifies the label on the left of the center space.  '(204 .
;; LEFT)' starts the first line of the list item with the same indentation as normal text.  Both measurements are measured in points from the left page
;; margin.

(RPAQ? IM.LABELED.LIST.TABS '(18 (186 . RIGHT)
                                 (204 . LEFT)))

;; left, right, top, and bottom margins of the 'page region' , measured in points from the four edges of the page.

(RPAQ? IM.PAGE.LEFTMARGIN 58)

(RPAQ? IM.PAGE.RIGHTMARGIN 54)

(RPAQ? IM.PAGE.TOPMARGIN 54)

(RPAQ? IM.PAGE.BOTTOMMARGIN 54)

;; top margin of the page region for the first page of a chapter <where the first paragraph is the chapter title>, measured in points from the top page
;; edge.

(RPAQ? IM.PAGE.FIRST.TOPMARGIN 12)

;; top margin of the page region for the first page of the index, measured in points from the top page edge.  Note that in the case of the index, because it
;; uses two columns, the index title is implemented as a Tedit header, instead of as the first paragraph of the document.

(RPAQ? IM.INDEX.PAGE.FIRST.TOPMARGIN 144)

;; y-pos of lower-left corner of footer text, measured in points from the bottom page edge.

(RPAQ? IM.FOOTER.Y 22)

;; y-pos of the footer hrule, measured in points from the bottom page edge.

(RPAQ? IM.FOOTER.RULE.Y 30)

;; y-pos of lower-left corner of header text, measured in points from the bottom page edge.

(RPAQ? IM.HEADER.Y 761)

;; y-pos of the header hrule, measured in points from the bottom page edge.

(RPAQ? IM.HEADER.RULE.Y 757)

;; y-pos of lower-left corner of bottom draft message, measured in points from the bottom page edge.

(RPAQ? IM.DRAFT.MESSAGE.BOTTOM.Y 5)

;; y-pos of lower-left corner of top draft message, measured in points from the bottom page edge.

(RPAQ? IM.DRAFT.MESSAGE.TOP.Y 775)

;; x-pos of lower-left corner of both top and bottom draft messages, measured in points from the left page edge.

(RPAQ? IM.DRAFT.MESSAGE.X 200)

(FILESLOAD TEDIT IMTRAN HRULE IMINDEX)

(DEFINEQ
```

(TRANSLATE.DUMPOUT

```
  [LAMBDA (DUMPOUT.ARGS)                                    (* mjs "18-Sep-85 16:17")
```

(* * this function translates the DUMPOUT macro form into a PROGN form that calls a series of functions, such as DUMP.)

(* * the indentation code has been commented out --- will try indenting everything to same, unless specified otherwise with PARALOOKS)

```
    (PROG ((DUMPOUT.FORMS NIL)
           (DUMPOUT.UNDO NIL)
           COMM COMM.ARG)
          [while DUMPOUT.ARGS do (SELECTQ (SETQ COMM (pop DUMPOUT.ARGS))
                                          (NIL)
                                          ((CR TAB START.PARA DUMP.FOOTNOTES START.SUPER START.SUB END.SUPER END.SUB)
                                                                    (* just pass these atoms as commands to DUMP)
                                               (push DUMPOUT.FORMS (LIST 'DUMP.FORMAT (KWOTE COMM))))
                                          ((FLUSH.ARG TRIVIAL.ARG DUMP.ARG)
                                               (push DUMPOUT.FORMS (LIST COMM)))
                                          (INDENT
```

(* * SELECTQ (SETQ COMM.ARG (pop DUMPOUT.ARGS)) (INIT
(push DUMPOUT.FORMS (QUOTE (PUT.MY.PROP (QUOTE INDENT) INITIAL.INDENT)))
(push DUMPOUT.FORMS (QUOTE (PUT.MY.PROP (QUOTE WIDTH) INITIAL.WIDTH)))
(push DUMPOUT.FORMS (QUOTE (DUMP.FORMAT (QUOTE INDENT) INITIAL.INDENT))))
(NONE (push DUMPOUT.FORMS (QUOTE (PUT.MY.PROP (QUOTE INDENT)
(QUOTE NONE)))) (push DUMPOUT.FORMS (QUOTE (PUT.MY.PROP
(QUOTE WIDTH) (ANC.WIDTH)))) (push DUMPOUT.FORMS (QUOTE
(DUMP.FORMAT (QUOTE INDENT) (QUOTE NONE))))) (push DUMPOUT.FORMS
(LIST (QUOTE (LAMBDA (I) (PUT.MY.PROP (QUOTE INDENT)
(IPLUS (ANC.INDENT) I)) (PUT.MY.PROP (QUOTE WIDTH) (IDIFFERENCE
(ANC.WIDTH) I)) (DUMP.FORMAT (QUOTE INDENT) (IPLUS
(ANC.INDENT) I)))) COMM.ARG)))

(* * push DUMPOUT.UNDO (QUOTE INDENT))

```
                                          (SETQ COMM.ARG (pop DUMPOUT.ARGS)))
                                          (WIDTH (push DUMPOUT.FORMS (LIST 'PUT.MY.PROP (KWOTE 'WIDTH)
                                                                           (pop DUMPOUT.ARGS))))
                                          (FONT (SETQ COMM.ARG (pop DUMPOUT.ARGS))
                                                [push DUMPOUT.FORMS (LIST 'DUMP.FORMAT (KWOTE 'FONT)
                                                                          (COND
                                                                             ((LISTGET IM.TEDIT.FONT.DEFS COMM.ARG)
                                                                              (KWOTE COMM.ARG))
                                                                             (T COMM.ARG]
                                                (push DUMPOUT.UNDO 'FONT))
                                          (PARALOOKS (push DUMPOUT.FORMS (LIST 'DUMP.FORMAT (KWOTE 'PARALOOKS)
                                                                               (pop DUMPOUT.ARGS))))
                                          (DUMP.CHARS (push DUMPOUT.FORMS (LIST (FUNCTION IM.DUMP.CHARS)
                                                                                (pop DUMPOUT.ARGS))))
                                          (push DUMPOUT.FORMS (LIST 'DUMP.FORMAT (KWOTE 'TEXT)
                                                                    (LIST 'MAKE.SAVE COMM]
          [for X in DUMPOUT.UNDO do (push DUMPOUT.FORMS (LIST 'DUMP.FORMAT (KWOTE 'UNDO)
                                                              (KWOTE X]
```

(* * push DUMPOUT.FORMS (QUOTE (PUT.MY.PROP (QUOTE INDENT) DUMPOUT.SAVE.INDENT)))

(* * push DUMPOUT.FORMS (QUOTE (PUT.MY.PROP (QUOTE WIDTH) DUMPOUT.SAVE.WIDTH)))

(* * RETURN (APPEND (QUOTE (PROG ((DUMPOUT.SAVE.INDENT
(GET.MY.PROP (QUOTE INDENT))) (DUMPOUT.SAVE.WIDTH
(GET.MY.PROP (QUOTE WIDTH)))))) (DREVERSE DUMPOUT.FORMS)))

```
          (RETURN (CONS 'PROGN (DREVERSE DUMPOUT.FORMS])
```

(TRANSLATE.SAVE.DUMPOUT

```
  [LAMBDA (SAVE.DUMPOUT.ARGS)                               (* mjs "12-Jan-84 15:00")
    (LSUBST SAVE.DUMPOUT.ARGS 'XXX '(PROG ((GOBBLE.SAVE.CONC (CONS)))
                                          (DECLARE (SPECVARS GOBBLE.SAVE.CONC))
                                          (DUMPOUT XXX)
                                          (RETURN GOBBLE.SAVE.CONC])

)

(DECLARE%: EVAL@COMPILE

(PUTPROPS IM.HOLD.FOOTNOTES MACRO [X `(PROG NIL
                                            (PUT.MY.PROP 'PASSFOOT T)
                                      %,@
                                      X     (PUT.MY.PROP 'PASSFOOT NIL)
                                            (DUMPOUT CR CR DUMP.FOOTNOTES])

(PUTPROPS DUMPOUT MACRO (X (TRANSLATE.DUMPOUT X)))

(PUTPROPS SAVE.DUMPOUT MACRO (X (TRANSLATE.SAVE.DUMPOUT X)))
)
```

FUNCTION INDEX

VARIABLE INDEX

PROPERTY INDEX

MACRO INDEX