```
(RPAQQ DOCUMENTCOMS
       ((* * This program creates documentation for any Lisp package in the style of the Lisp Users
         documentation. It requires that the code for the package be loaded. It does not completely format the
         document, but it minimizes the amount of work that the documenter must do. To make most effective use
         of these functions, each function in the package being documented must have a comment as the first
         expression in the function after the timestamp. Similarly, the COMS variable of the file should
         contain a comment (like this one)
         as the first item in the list. This package was documented using itself.)
        (FNS Document.Create Document.Begin Document.FileComments Document.Functions Document.Variables
           Document.Things Document.Finish Document.RunningHead Document.Title Document.Information
           Document.SectionHead Document.Format Document.FunctionCommentedP)))
```

(* * This program creates documentation for any Lisp package in the style of the Lisp Users documentation.
It requires that the code for the package be loaded. It does not completely format the document, but it minimizes the amount
of work that the documenter must do. To make most effective use of these functions, each function in the package being
documented must have a comment as the first expression in the function after the timestamp.
Similarly, the COMS variable of the file should contain a comment
(like this one) as the first item in the list. This package was documented using itself.)

```
(DEFINEQ
```

(Document.Create
```
  [LAMBDA (FileName)                                        (* Newman "29-Aug-86 10:07")
```

(* * This function builds a document for a loaded file. The document is in the style of the Lisp Library package
documentation. The function collects comments from the COMS variable of the file and from the functions in the file.
It also collects some information from the Interlisp-D file package.
This is the top-level function in the ODCUMENT package.)

```
    (if (MEMBER FileName FILELST)
        then (RESETLST
                 (RESETSAVE (CURSOR WAITINGCURSOR))
                 (LET ((Stream (OPENTEXTSTREAM))
                        Pointer)
                      (Document.Begin FileName Stream)
                      (SETQ Pointer (ADD1 (GETEOFPTR Stream)))
                      (Document.FileComments FileName Stream)
                      (Document.Functions FileName Stream)
                      (Document.Variables FileName Stream)
                      (Document.Things FileName Stream (QUOTE MACROS)
                             (QUOTE Macro))
                      (Document.Things FileName Stream (QUOTE RECORDS)
                             (QUOTE Record))
                      (Document.Finish Stream Pointer)))
      else (ERROR FileName " not a loaded file."])
```

(Document.Begin
```
  [LAMBDA (FileName Stream)                                 (* Newman "29-Aug-86 10:23")
```

(* * This function initializes the begining of the document stream.)

```
    (Document.RunningHead FileName Stream)
    (Document.Title FileName Stream (ADD1 (GETEOFPTR Stream)))
    (Document.Information FileName Stream (ADD1 (GETEOFPTR Stream)])
```

(Document.FileComments
```
  [LAMBDA (FileName Stream)                                 (* Newman "29-Aug-86 10:17")
```

(* * This function places all comments found in the COMS variable of FileName into Stream.
It is intended for use in documenting a Lisp code file. Ideally, these commends would be distributed through the
documentation as they are distributed through the COMS variable.)

```
    (Document.SectionHead Stream "INTRODUCTION")
    (PROMPTPRINT "Collecting File Comments ...")
    (for Descriptor in (FILECOMSLST FileName (QUOTE *)) do (printout Stream .PPFTL (REMOVE (QUOTE *)
                                                                                    Descriptor)
                                                              T])
```

(Document.Functions

```
  [LAMBDA (FileName Stream)                                              (* Newman "29-Aug-86 10:25")

          (* * This function documents the Functions on FileName into Stream.
          It does this by obtaining the function names from the file package, using the ARGLIST function to obtain the argument list,
          and obtainin the initial comment in the function if it exists.)

    (Document.SectionHead Stream "FUNCTIONS")
    (PROMPTPRINT "Collecting Function Comments ... ")
    (for Function in (FILEFNSLST FileName)
       do (TEDIT.INSERT Stream (CONCAT "(" Function " ")
                   (ADD1 (GETEOFPTR Stream))
                   (FONTCREATE (QUOTE MODERN)
                           10
                           (QUOTE MRR)))
          (TEDIT.INSERT Stream (SUBSTRING (MKSTRING (ARGLIST Function))
                                    2 -2)
                   (ADD1 (GETEOFPTR Stream))
                   (FONTCREATE (QUOTE MODERN)
                           10
                           (QUOTE ITALIC)))
          (TEDIT.INSERT Stream (CONCAT ")        [Function]" (CHARACTER 13))
                   (ADD1 (GETEOFPTR Stream))
                   (FONTCREATE (QUOTE MODERN)
                           10
                           (QUOTE MRR)))

          (* * This SETFILEPTR is here because someone leaves the FILEPTR in the wrong place.)

          (SETFILEPTR Stream (GETEOFPTR Stream))
          (if (Document.FunctionCommentedP Function)
              then (printout Stream .PPVTL [REMOVE (QUOTE *)
                                              (CADDDR (OR (GETPROP Function (QUOTE EXPR))
                                                      (GETD Function]
                       T])
```

(**Document.Variables**

```
  [LAMBDA (FileName Stream)                                              (* Newman "29-Aug-86 10:28")

          (* This function is intended for use while documenting Lisp code.
          It collects a list of the variables from FileName, and places them in Stream in the format of the Lisp Library Documentation.)

    (Document.SectionHead Stream "VARIABLES")
    (PROMPTPRINT "Collecting Variables ...")
    (for Variable in (FILECOMSLST FileName (QUOTE VARS))
       do (if (MEMBER Variable (FILECOMSLST FileName (QUOTE GLOBALVARS)))
              then (printout Stream Variable , "        [GlobalVar]" T)
            elseif (MEMBER Variable (FILECOMSLST FileName (QUOTE SPECVARS)))
              then (printout Stream Variable , "        [SpecialVar]" T)
            else (printout Stream Variable , "   [Variable]" T))
```

(**Document.Things**

```
  [LAMBDA (FileName Stream FilePkgType TypeName)                         (* Newman "29-Aug-86 10:32")

          (* This function is intended for use while documenting Lisp code.
          It collects a list of items of type TypeName, that are written using the FilePkgType file package command, from FileName,
          and places them in Stream in the format of the Lisp Library Documentation.)

    (if (AND FilePkgType (MEMBER FilePkgType FILEPKGTYPES))
        then (if (FILECOMSLST FileName FilePkgType)
                 then (Document.SectionHead Stream (CONCAT (U-CASE (SETQ TypeName (OR TypeName FilePkgType)))
                                                      "S"))
                      (PROMPTPRINT (CONCAT "Collecting " TypeName " ..."))
                      (for Thing in (FILECOMSLST FileName FilePkgType) do (printout Stream Thing ,
                                                                  (CONCAT (CHARACTER 9)
                                                                      "[" TypeName "]")
                                                           T)))
        else (ERROR FilePkgType "Bad file package type: "])
```

(**Document.Finish**

```
  [LAMBDA (Stream Pointer)                                               (* Newman "28-Aug-86 15:22")

          (* * This function performs formatting and the like to make the document pretty, and to make the format correct as specified.)

          (* * Set the selection to cover the non-header portions of the file.)

    (TEDIT.SETSEL Stream Pointer (DIFFERENCE (GETEOFPTR Stream)
                                       Pointer)
              (QUOTE LEFT))

          (* * Eliminate extra spaces)

    (bind (Num _ 1) repeatuntil (ZEROP Num) do (SETQ Num (TEDIT.SUBSTITUTE Stream "  " " ")))

          (* * Eliminate carriage returns that I think are introduced by printout)
```

```
(TEDIT.SUBSTITUTE Stream (CONCAT (CHARACTER 13)
                                      " ")
     " ")
```

(* * For some reason, the paragraph breaks don't work unless this line is here.)

```
(TEDIT.SUBSTITUTE Stream (MKSTRING (CHARACTER 13))
       (MKSTRING (CHARACTER 13)))
```

(* * Set the page, paragraph, and character looks of the non-header portions of the document.)

```
(Document.Format Stream)
```

(* * TEdit the stream so the user can finish the job by hand.)

```
(TEDIT Stream])
```

(Document.RunningHead
```
[LAMBDA (FileName Stream)                                                    (* Newman "28-Aug-86 15:57")
```

(* * This function creates the running header for the document.)

```
(TEDIT.INSERT Stream "XEROX" 0 (FONTCREATE (QUOTE LOGO)
                                               24))
(TEDIT.INSERT Stream (CONCAT " " (CHARACTER 9)
                               FileName
                               (CHARACTER 13)
                               (CHARACTER 13))
       NIL
       (FONTCREATE (QUOTE MODERN)
             10))
(TEDIT.INSERT.OBJECT (HRULE.CREATE (QUOTE 2))
       Stream
       (GETEOFPTR Stream))
(TEDIT.PARALOOKS (TEXTOBJ Stream)
       (QUOTE (TYPE PAGEHEADING SUBTYPE RUNNINGHEAD RIGHTMARGIN 456 LEFTMARGIN 0 TABS (NIL (456 . RIGHT))
                    QUAD JUSTIFIED))
       0
       (ADD1 (GETEOFPTR Stream)))
```

(* * we set the file ptr to be kind to others, as TEDIT.PARALOOKS moves it from the end of the file.)

```
(SETFILEPTR Stream (GETFILEPTR Stream])
```

(Document.Title
```
[LAMBDA (FileName Stream Pointer)                                            (* Newman "29-Aug-86 11:35")
```

(* * This function creates the title area of the document consisting of the name of the package being documented in between two lines.)

```
(TERPRI Stream)
```

(* This TERPRI is here because otherwise the first HRULE in the title becomes a part of the running header.
This happens for no apparent reason, and is a mystery to me.)

```
(TEDIT.INSERT.OBJECT (HRULE.CREATE (QUOTE (5 4 1)))
       Stream
       (ADD1 (GETEOFPTR Stream)))
(TEDIT.INSERT Stream (CONCAT (CHARACTER 13)
                               FileName
                               (CHARACTER 13)
                               (CHARACTER 13))
       (ADD1 (GETEOFPTR Stream))
       (FONTCREATE (QUOTE MODERN)
             12
             (QUOTE BRR)))
(TEDIT.INSERT.OBJECT (HRULE.CREATE (QUOTE (1 4 5)))
       Stream
       (GETEOFPTR Stream))
(TEDIT.PARALOOKS (TEXTOBJ Stream)
       (QUOTE (TYPE NIL PARALEADING 6 LINELEADING 4 RIGHTMARGIN 312 LEFTMARGIN 138 1STLEFTMARGIN 138 QUAD
                    CENTERED))
       Pointer
       (DIFFERENCE (ADD1 (GETEOFPTR Stream))
             Pointer))
```

(* * This call to SETFILEPTR is here because I believe TEDIT.PARALOOKS does not leave the file pointer at the end of the stream as the printout function requires. I believe that if I eliminate all calls to printout, I can eliminate all calls to SETFILEPTR. I also believe that PRINTOUT can always be replaced by TEDIT.INSERT.)

```
(SETFILEPTR Stream (GETEOFPTR Stream])
```

(Document.Information

```
[LAMBDA (FileName Stream Pointer)                                        (* Newman "29-Aug-86 10:37")

        (* * This function creates the information at the top of the document, including the form for the author's name and a list of
        other packages necessary to run this package.)

    (TEDIT.INSERT Stream (CONCAT " By: >>Author's Name<< (>>Net Address<<)" (CHARACTER 13)
                                " "
                                (SUBSTRING (DATE)
                                        1 9)
                                (CHARACTER 13))
            (ADD1 (GETEOFPTR Stream))
            (FONTCREATE (QUOTE MODERN)
                    10)
            T)
    (if (FILECOMSLST FileName (QUOTE FILES))
        then (TEDIT.INSERT Stream (CONCAT "The following packages are loaded by " FileName ": "
                                        (SUBSTRING (MKSTRING (FILECOMSLST FileName (QUOTE FILES)))
                                                2 -2)
                                        (CHARACTER 13)
                                        " ")
                (ADD1 (GETEOFPTR Stream))
                (FONTCREATE (QUOTE MODERN)
                        10)
                T))
    (TEDIT.PARALOOKS (TEXTOBJ Stream)
        (QUOTE (RIGHTMARGIN 456 LEFTMARGIN 0 1STLEFTMARGIN 0 QUAD CENTERED PARALEADING 17))
        Pointer
        (ADD1 (GETEOFPTR Stream)))

        (* The following SETFILEPTR exists because TEDIT.PARALOOKS does not leave the file pointer at the end of the file.)

    (SETFILEPTR Stream (GETEOFPTR Stream)]
```

(**Document.SectionHead**
```
  [LAMBDA (Stream String)                                               (* Newman "29-Aug-86 10:39")

        (* * This function is intended to create the bold section headings in Stream that are needed for the Lisp Users document
        style.)

    (TEDIT.INSERT Stream (CONCAT String (CHARACTER 13))
        (ADD1 (GETEOFPTR Stream))
        (FONTCREATE (QUOTE MODERN)
                10
                (QUOTE BRR)))
    (SETFILEPTR Stream (GETEOFPTR Stream))

        (* * The following TEDIT.CARETLOOKS prevents the bolding from continuing past the section heading in some cases.)

    (TEDIT.CARETLOOKS Stream (FONTCREATE (QUOTE MODERN)
                                    10
                                    (QUOTE MRR])
```

(**Document.Format**
```
  [LAMBDA (Stream)                                                      (* Newman "29-Aug-86 10:54")

        (* * This function formats the document. It sets the page layout, font looks, and paragraph formatting for all selected text.)

    (TEDIT.SUBLOOKS Stream (QUOTE (FAMILY GACHA))
        (QUOTE (FAMILY MODERN SIZE 10)))
    (TEDIT.PARALOOKS (TEXTOBJ Stream)
        (QUOTE (TABS (NIL (0 . RIGHT)
                        (456 . RIGHT))
                LINELEADING 4 PARALEADING 11 RIGHTMARGIN 456 LEFTMARGIN 0 1STLEFTMARGIN 0 QUAD JUSTIFIED
                ))
        (TEDIT.GETSEL Stream))
    (TEDIT.PAGEFORMAT Stream (TEDIT.COMPOUND.PAGEFORMAT (TEDIT.SINGLE.PAGEFORMAT T 26.5 3.0 (QUOTE (MODERN 10))
                                                            (QUOTE CENTERED)
                                                            7.0 6.0 8.0 8.0 1 38.0 0
                                                            (QUOTE ((RUNNINGHEAD 7.0 62.0)))
                                                            (QUOTE PICAS)
                                                            (QUOTE (STARTINGPAGE# 1))
                                                            (QUOTE LETTER))
                                (TEDIT.SINGLE.PAGEFORMAT T 26.5 3.5 (QUOTE (MODERN 10))
                                        (QUOTE CENTERED)
                                        7.0 6.0 8.0 8.0 1 38.0 0 (QUOTE ((RUNNINGHEAD 7.0 62.0)))
                                        (QUOTE PICAS)
                                        (QUOTE (STARTINGPAGE# 1))
                                        (QUOTE LETTER))
                                (TEDIT.SINGLE.PAGEFORMAT T 26.5 3.0 (QUOTE (MODERN 10))
                                        (QUOTE CENTERED)
                                        7.0 6.0 8.0 8.0 1 38.0 0 (QUOTE ((RUNNINGHEAD 7.0 62.0)))
                                        (QUOTE PICAS)
                                        (QUOTE (STARTINGPAGE# 1))
                                        (QUOTE LETTER]))
```

₍**Document.FunctionCommentedP**
```
  [LAMBDA (Function)                                                    (* Newman "29-Aug-86 10:42")

          (* * This function is intended to tell if a function has an initial comment or not.)

      (EQUAL (QUOTE *)
             (CAR (CADDDR (OR (GETPROP Function (QUOTE EXPR))
                              (GETD Function)
                              (ERROR Function "Not a function: "])

)
```

FUNCTION INDEX

VARIABLE INDEX