File created: 15-Feb-89 15:42:25 **{DSK}/pooh/pedersen/lisp/XCL-BRIDGE.;2**

changes to: (IL:VARS IL:XCL-BRIDGECOMS)
(IL:VARIABLES *BRIDGING*)
(IL:FUNCTIONS MANAGED-TO-TEXT-FILE TEXT-TO-MANAGED-FILE)

previous date: 6-Dec-88 17:22:36 {DSK}/pooh/pedersen/lisp/XCL-BRIDGE.;1

Read Table: XCL

Package: XEROX-COMMON-LISP

Format: XCCS

; Copyright (c) 1988, 1989 by Xerox Corporation. All rights reserved.

```
(IL:RPAQQ IL:XCL-BRIDGECOMS
          ((IL:DECLARE\: IL:DOCOPY IL:DONTEVAL@LOAD IL:DONTEVAL@COMPILE (IL:P (EXPORT '(TEXT-TO-MANAGED-FILE
                                                                                         MANAGED-TO-TEXT-FILE
                                                                                         *BRIDGING*)
                                                                                       (FIND-PACKAGE "XCL"))))
           (IL:COMS
                  ;; indicator free variable
                  (IL:VARIABLES *BRIDGING*))
           (IL:COMS
                  ;; From Text  to manager format
                  (IL:VARIABLES *EOF-MARKER*)
                  (IL:FUNCTIONS TEXT-TO-MANAGED-FILE)
                  (IL:FUNCTIONS CONSTRUCT-COMS INSTALL-FILE)
                  (IL:FUNCTIONS READ-SEMICOLON-COMMENT MAKE-SEMICOLON-COMMENT PROBE-FOR-MODE-LINE)
                  (IL:FUNCTIONS COMBINE-COMMENTS COMMENT-P COMMENT-COMBINEABLE-P DO-COMBINE-COMMENTS)
                  (IL:FUNCTIONS PROCESS-DEFINITIONS DEFINER-TYPE FIND-DEFINITION))
           (IL:COMS
                  ;; From manager to text format
                  (IL:FUNCTIONS MANAGED-TO-TEXT-FILE)
                  (IL:FUNCTIONS CONSTRUCT-MODE-LINE GET-COMS-FORMS MAKE-COMMENT))
           (FILE-ENVIRONMENTS "XCL-BRIDGE")
           (IL:COMS
                  ;; comment identity preservation hack
                  (IL:VARIABLES *PRESERVE-COMMENT-START-CHAR* *PRESERVE-COMMENT-START-CHARCODE*)
                  (IL:FUNCTIONS INITIAL-COMMENT-LINE-P FIX-COMMENT-?)
                  (IL:ADVICE (IL:CONCAT :IN IL:PRIN2-LONG-STRING)
                         (IL:PRIN1 :IN IL:PRIN2-LONG-STRING)))))

(IL:DECLARE\: IL:DOCOPY IL:DONTEVAL@LOAD IL:DONTEVAL@COMPILE

(EXPORT '(TEXT-TO-MANAGED-FILE MANAGED-TO-TEXT-FILE *BRIDGING*)
        (FIND-PACKAGE "XCL"))
)

;; indicator free variable

(DEFVAR *BRIDGING* NIL
   "True while dynamically within the XCL-BRIDGE")

;; From Text  to manager format

(DEFPARAMETER *EOF-MARKER* "eof")

(DEFUN TEXT-TO-MANAGED-FILE (PATHNAME FILENAME &KEY (PACKAGE "USER" PACKAGE-P)
                                     (READTABLE "XCL" READTABLE-P)
                                     (READ-BASE 10 READ-BASE-P)
                                     (COMPILER :COMPILE-FILE)
                                     (COMBINE-COMMENTS-P T))
   (PROG ((ROOTNAME (INTERN (STRING FILENAME)
                            (FIND-PACKAGE "INTERLISP")))
          FORMS FIRST-FORM COMS)
         (WITH-OPEN-FILE (STREAM PATHNAME :DIRECTION :INPUT)
                (MULTIPLE-VALUE-SETQ (PACKAGE READTABLE READ-BASE FIRST-FORM)
                       (PROBE-FOR-MODE-LINE STREAM PACKAGE PACKAGE-P READTABLE READTABLE-P READ-BASE
                              READ-BASE-P))
                ;; Declare read environment
                (FORMAT T "Using the following read environment:~%Package: ~a Readtable: ~a Read-base: ~a~%"
                       PACKAGE READTABLE READ-BASE)
                (UNLESS (Y-OR-N-P "Do you wish to continue? ")
                       (RETURN NIL))
                (LET ((*PACKAGE* (FIND-PACKAGE PACKAGE))
                      (*READTABLE* (COPY-READTABLE (IL:FIND-READTABLE READTABLE)))
                      (*READ-BASE* READ-BASE)
```

```
                    (*BRIDGING* T))
                   ;; Setup for reading comments properly
                   (SET-MACRO-CHARACTER #\; 'READ-SEMICOLON-COMMENT NIL *READTABLE*)
                   (SETQ FORMS (WITH-COLLECTION
                                 (DO ((FORM (READ STREAM NIL *EOF-MARKER*)
                                            (READ STREAM NIL *EOF-MARKER*)))
                                     ((EQ FORM *EOF-MARKER*))
                                   (UNLESS PACKAGE-P
                                       (IF (EQ (CAR FORM)
                                               'IN-PACKAGE)
                                           (LET ((NEW-PACKAGE-NAME (STRING (EVAL (SECOND FORM)))))
                                                (WHEN (NOT (STRING= NEW-PACKAGE-NAME PACKAGE))
                                                    (WARN "*** Encountered in-package form: Changing to ~a
                                                          package" NEW-PACKAGE-NAME)
                                                    (SETQ PACKAGE (PACKAGE-NAME (EVAL FORM)))))))
                                   (COLLECT FORM))))
                   (IF FIRST-FORM
                       (SETQ FORMS (CONS FIRST-FORM FORMS)))
                   (WHEN COMBINE-COMMENTS-P
                       (FORMAT T "Combining comments..~%")
                       (SETQ FORMS (COMBINE-COMMENTS FORMS)))))
          (WHEN (Y-OR-N-P "Edit the forms read prior to constructing a coms list? ")
              (SEDIT:SEDIT FORMS)
              (UNLESS (Y-OR-N-P "Do you wish to continue? ")
                      (RETURN NIL)))
          (SETQ COMS (CONSTRUCT-COMS FORMS))
          (WHEN (Y-OR-N-P "Edit the coms prior to installing the file? ")
              (SEDIT:SEDIT COMS)
              (UNLESS (Y-OR-N-P "Do you wish to continue? ")
                      (RETURN NIL)))
          (WHEN (Y-OR-N-P "Install file? ")
              (RETURN (INSTALL-FILE ROOTNAME COMS FORMS :PACKAGE PACKAGE :READTABLE READTABLE :READ-BASE READ-BASE
                            :COMPILER COMPILER)))))

(DEFUN CONSTRUCT-COMS (FORMS)
    ;; Constructs a file coms expression for a list of top-level forms
    (LET ((COMS NIL)
          (CURRENT-DEFINITIONS NIL)
          (CURRENT-TYPE :NONE)
          (DEFINER-TYPE NIL))
         (DOLIST (FORM FORMS)
             (SETQ DEFINER-TYPE (DEFINER-TYPE FORM))
             (WHEN (AND (NOT (EQ CURRENT-TYPE DEFINER-TYPE))
                        CURRENT-DEFINITIONS)
                 (SETQ COMS (PROCESS-DEFINITIONS CURRENT-DEFINITIONS CURRENT-TYPE COMS))
                 (SETQ CURRENT-DEFINITIONS NIL CURRENT-TYPE :NONE))
             (COND
                ((EQ DEFINER-TYPE :EVAL-WHEN)
                 (SETQ COMS (NCONC COMS `((EVAL-WHEN ,(CADR FORM)
                                                     ,@(CONSTRUCT-COMS (CDDR FORM)))))))
                (T (SETQ CURRENT-TYPE DEFINER-TYPE)
                   (PUSH FORM CURRENT-DEFINITIONS))))
         (WHEN CURRENT-DEFINITIONS
             (SETQ COMS (PROCESS-DEFINITIONS CURRENT-DEFINITIONS CURRENT-TYPE COMS)))
        COMS))

(DEFUN INSTALL-FILE (NAME COMS FORMS &KEY (PACKAGE "USER")
                                          (READTABLE "XCL")
                                          (READ-BASE 10)
                                          (COMPILER :COMPILE-FILE))
   (LABELS ((INSTALL-DEFINITIONS (COMS FORMS)
                  (DOLIST (FORM FORMS)
                      (LET ((DEF-TYPE (DEFINER-TYPE FORM))
                            NAME)
                           (COND
                              ((EQ DEF-TYPE :EVAL-WHEN)
                               (INSTALL-DEFINITIONS COMS (CDDR FORM)))
                              ((AND DEF-TYPE (NOT (EQ DEF-TYPE :COMMENT)))
                               (SETQ NAME (%DEFINER-NAME (CAR FORM)
                                                         (REMOVE-COMMENTS FORM)))
                               (WHEN (FIND-DEFINITION NAME DEF-TYPE COMS)
                                   ;; Save Definition
                                   (%DEFINE-TYPE-SAVE-DEFN NAME DEF-TYPE FORM))))))))
          (SETQ FORMS (NCONC FORMS `((DEFINE-FILE-ENVIRONMENT ,(STRING NAME) :PACKAGE ,PACKAGE
                                         :READTABLE ,READTABLE
                                         :BASE ,READ-BASE
                                         :COMPILER ,COMPILER))))
          (SETQ COMS (NCONC COMS `((FILE-ENVIRONMENTS ,(STRING NAME)))))
          (INSTALL-DEFINITIONS COMS FORMS)
          (LET ((ROOT-NAME (INTERN (STRING NAME)
                                   (FIND-PACKAGE "INTERLISP"))))
               (SET (IL:FILECOMS ROOT-NAME)
```

```
                COMS)
            (IL:ADDFILE NAME)
            (SETF (GET ROOT-NAME 'IL:FILETYPE)
                  COMPILER)
            (SETF (GET ROOT-NAME 'IL:MAKEFILE-ENVIRONMENT)
                  `(:READTABLE ,READTABLE :PACKAGE ,PACKAGE :BASE ,READ-BASE))
            ROOT-NAME)))
```

```
(DEFUN READ-SEMICOLON-COMMENT (STREAM &OPTIONAL DISP-CHAR)
  ;; Adjacent comments of the same level are smashed together during an after-read pass over the structure.
  (DECLARE (IGNORE DISP-CHAR))
  (LET ((LEVEL (LET ((VALUE 0)
                     (CH NIL))
                 (LOOP (WHEN (NOT (EQL (SETQ CH (READ-CHAR STREAM))
                                       #\;))
                         (UNREAD-CHAR CH STREAM)
                         (RETURN VALUE))
                       (INCF VALUE))))))
    (MAKE-SEMICOLON-COMMENT (READ-LINE STREAM)
          LEVEL)))
```

```
(DEFUN MAKE-SEMICOLON-COMMENT (STRING LEVEL)
  `                                                                    (IL:* (IL:\\\, (CDR (ASSOC (MOD LEVEL 3)
                                                                       (QUOTE ((0 . IL:\;) (1 . IL:|;;|) (2 . IL:|;;;|))) :TEST
                                                                       (FUNCTION EQ)))) (IL:\\\, (STRING-TRIM
                                                                       (QUOTE (#\Space #\Tab)) STRING))))
```

```
(DEFUN PROBE-FOR-MODE-LINE (STREAM PACKAGE PACKAGE-P READTABLE READTABLE-P READ-BASE READ-BASE-P)
                                                                       ; Edited 4-Aug-88 15:13 by ht:
  (LET*
   ((MODE-FORM (DO ((CH (READ-CHAR STREAM)
                        (READ-CHAR STREAM)))
                   ((NOT (MEMBER CH '(#\Space #\Newline #\Tab)
                                 :TEST
                                 #'EQ))
                    (UNREAD-CHAR CH STREAM)
                    (IF (EQ CH #\;)
                        (READ-SEMICOLON-COMMENT STREAM)))))
    (MODE-STRING (AND MODE-FORM (STRING= (THIRD MODE-FORM)
                                         "-*-" :END1 3)
                      (STRING-UPCASE (THIRD MODE-FORM)
                            :START 3))))
   (WHEN MODE-STRING
       (LET ((PACKAGE-MARKER "PACKAGE:")
             (SYNTAX-MARKER "SYNTAX:")
             (BASE-MARKER "BASE:")
            MODE-POSITION MODE-NAME)
           (WHEN (AND (NULL PACKAGE-P)
                      (SETQ MODE-POSITION (SEARCH PACKAGE-MARKER MODE-STRING)))
               (LET* ((PACKAGE-FORM (READ-FROM-STRING MODE-STRING NIL NIL :START (+ MODE-POSITION (LENGTH
                                                                                                  PACKAGE-MARKER
                                                                                                  ))))
                      (PACKAGE-NAME (STRING (IF (CONSP PACKAGE-FORM)
                                                (CAR PACKAGE-FORM)
                                                PACKAGE-FORM))))
                     (SETQ PACKAGE (IF (FIND-PACKAGE PACKAGE-NAME)
                                       PACKAGE-NAME
                                       (PROGN (CERROR "Create it and carry on" "~&Non-existent package: ~S~%"
                                                      PACKAGE-NAME)
                                              (IF (CONSP PACKAGE-FORM)
                                                  (LET ((USE-LIST (OR (SECOND (MEMBER :USE PACKAGE-FORM :TEST
                                                                                      #'EQ))
                                                                      (SECOND PACKAGE-FORM)))
                                                        (NICKNAMES (SECOND (MEMBER :NICKNAMES PACKAGE-FORM
                                                                                   :TEST #'EQ))))
                                                    (MAKE-PACKAGE PACKAGE-NAME :USE (OR USE-LIST "LISP")
                                                          :NICKNAMES NICKNAMES))
                                                  (MAKE-PACKAGE PACKAGE-NAME)))))))
           (WHEN (PACKAGEP PACKAGE)
               (SETQ PACKAGE (STRING (PACKAGE-NAME PACKAGE))))
           (WHEN (AND (NULL READTABLE-P)
                      (SETQ MODE-POSITION (SEARCH SYNTAX-MARKER MODE-STRING)))
               (SETQ MODE-NAME (STRING (READ-FROM-STRING MODE-STRING NIL NIL :START (+ MODE-POSITION
                                                                                       (LENGTH SYNTAX-MARKER))
                                                         )))
               (IF (OR (STRING= MODE-NAME "LISP")
                       (STRING= MODE-NAME "COMMON-LISP"))
                   (SETQ MODE-NAME "XCL"))
               (IF (NOT (IL:FIND-READTABLE MODE-NAME))
                   (ERROR "~&Non-existent readtable: ~A~%" MODE-NAME))
               (SETQ READTABLE MODE-NAME))
           (WHEN (READTABLEP READTABLE)
```

```
                (LET ((NAME (IL:READTABLEPROP READTABLE 'IL:NAME)))
                     (IF (NULL NAME)
                         (ERROR "Readtable ~s has no name." READTABLE)
                         (SETQ READTABLE NAME))))
            (WHEN (AND (NULL READ-BASE-P)
                       (SETQ MODE-POSITION (SEARCH BASE-MARKER MODE-STRING)))
                (SETQ MODE-NAME (READ-FROM-STRING MODE-STRING NIL NIL :START (+ MODE-POSITION (LENGTH
                                                                                                BASE-MARKER
                                                                                                ))))
                (IF (NOT (AND (NUMBERP MODE-NAME)
                              (> MODE-NAME 0)))
                    (ERROR "~&Bad read base: ~A~%" MODE-NAME))
                (SETQ READ-BASE MODE-NAME))))
    (VALUES PACKAGE READTABLE READ-BASE
```

;; Return a non-mode line comment, if necessary

```
            (AND (NULL MODE-STRING)
                 MODE-FORM))))
```

(DEFUN **COMBINE-COMMENTS** (X) ; Edited 10-Aug-88 10:19 by ht:

;;; Smash together adjacent sedit comments at the same level.

```
   (COND
      ((NOT (CONSP X))
       X)
      ((AND (COMMENT-P (CAR X))
            (COMMENT-P (CADR X))
            (COMMENT-COMBINEABLE-P (CAR X)
                   (CADR X)))
```

;; At least two adjacent comments at the same level

```
       (LET ((TAIL (CDDR X))
             (MATCHER (CADR (CAR X)))
             (COMMENTS (LIST (CAR X)
                             (CADR X))))
            (NCONC COMMENTS (WITH-COLLECTION (LOOP (IF (NOT (AND (COMMENT-P (CAR TAIL))
                                                                 (EQ (CADR (CAR TAIL))
                                                                     MATCHER)
                                                                 (NOT (INITIAL-COMMENT-LINE-P (CADDR (CAR TAIL)))
                                                                      )))
                                                       (RETURN NIL))
                                                   (COLLECT (CAR TAIL))
                                                   (SETQ TAIL (CDR TAIL)))))
            (FIX-COMMENT-? (CAR COMMENTS))
            (CONS (DO-COMBINE-COMMENTS COMMENTS MATCHER)
                  (COMBINE-COMMENTS TAIL))))
      (T (FIX-COMMENT-? X)
         (LET ((A (COMBINE-COMMENTS (CAR X)))
               (D (COMBINE-COMMENTS (CDR X))))
              (IF (AND (EQ A (CAR X))
                       (EQ D (CDR X)))
                  X
                  (CONS A D))))))
```

```
(DEFUN COMMENT-P (FORM)
   (AND (CONSP FORM)
        (EQ (CAR FORM)
            'IL:*)
        (CONSP (CDR FORM))
        (MEMBER (CADR FORM)
                '(IL:\; IL:|;;| IL:|;;;|)
                :TEST
                #'EQ)
        T))
```

```
(DEFUN COMMENT-COMBINEABLE-P (C1 C2) ; Edited 10-Aug-88 10:19 by ht:
   (AND (EQ (CADR C1)
            (CADR C2))
        (NOT (INITIAL-COMMENT-LINE-P (CADDR C2)))))
```

```
(DEFUN DO-COMBINE-COMMENTS (COMMENTS LEVEL)
```

;; COMMENTS is a list of sedit like comments at the same level

```
   `
```

```
(IL:* (IL:\\\, LEVEL) (IL:\\\, (APPLY
(QUOTE CONCATENATE) (QUOTE STRING)
(WITH-COLLECTION (DOLIST (COMMENT COMMENTS)
(LET ((STRING (THIRD COMMENT)))
(WHEN (> (LENGTH STRING) 0)
(COLLECT STRING) (COLLECT " "))))))))
```

```
(DEFUN PROCESS-DEFINITIONS (DEFINITIONS TYPE COMS)
```

```
   (CASE TYPE
       (:COMMENT (NCONC COMS (NREVERSE DEFINITIONS)))
       ((NIL)
          ;; Untyped forms
          (NCONC COMS `((IL:P ,.(NREVERSE DEFINITIONS)))))
       (OTHERWISE
          ;; Typed definitions
          (NCONC COMS `((,TYPE ,.(LET ((NAMES NIL)
                                       DEF)
                                      (LOOP (IF (NULL (SETQ DEF (POP DEFINITIONS)))
                                                (RETURN NAMES))
                                            (PUSH (%DEFINER-NAME (CAR DEF)
                                                                 (REMOVE-COMMENTS DEF))
                                                  NAMES)))))))))

(DEFUN DEFINER-TYPE (FORM)
   (COND
      ((COMMENT-P FORM)
       :COMMENT)
      ((AND (CONSP FORM)
            (SYMBOLP (CAR FORM))
            (OR (IF (EQ (CAR FORM)
                        'EVAL-WHEN)
                    :EVAL-WHEN)
                (GET (CAR FORM)
                     :DEFINER-FOR))))))

(DEFUN FIND-DEFINITION (NAME TYPE COMS)
   (DOLIST (EXPR COMS NIL)
       (LET ((FIRST (CAR EXPR)))
            (COND
               ((EQ FIRST TYPE)
                (IF (MEMBER NAME (CDR EXPR)
                            :TEST
                            'EQUAL)
                    (RETURN T)))
               ((EQ FIRST 'EVAL-WHEN)
                (IF (FIND-DEFINITION NAME TYPE (CDDR EXPR))
                    (RETURN T)))
               ((EQ FIRST 'IL:COMS)
                (IF (FIND-DEFINITION NAME TYPE (CDR EXPR))
                    (RETURN T)))))))

;; From manager to text format

(DEFUN MANAGED-TO-TEXT-FILE (FILENAME PATHNAME &KEY (PACKAGE "USER" PACKAGE-P)
                                     (READTABLE "LISP" READTABLE-P)
                                     (PRINT-BASE 10 PRINT-BASE-P)
                                     (LINELENGTH 72)
                                     (COMMENTS :PRESERVE))
   (LET ((ROOT-NAME (INTERN (STRING FILENAME)
                            (FIND-PACKAGE "INTERLISP")))
         MODE-LINE PACKAGE-FORM)
        (MULTIPLE-VALUE-SETQ (PACKAGE READTABLE PRINT-BASE MODE-LINE PACKAGE-FORM)
              (CONSTRUCT-MODE-LINE ROOT-NAME PACKAGE PACKAGE-P READTABLE READTABLE-P PRINT-BASE PRINT-BASE-P))
        (LET ((*BRIDGING* T)
              (*PACKAGE* (FIND-PACKAGE PACKAGE))
              (*READTABLE* (IL:FIND-READTABLE READTABLE))
              (*PRINT-BASE* PRINT-BASE)
              (*PRINT-CASE* :DOWNCASE)
              (*PRINT-ARRAY* T)
              (*PRINT-LEVEL* NIL)
              (*PRINT-LENGTH* NIL)
              (*PRINT-STRUCTURE* T)
              ;; Interlisp gorp that controls pretty printing
              (IL:*PRINT-SEMICOLON-COMMENTS* (OR COMMENTS T))
              (IL:FONTCHANGEFLG NIL)
              (IL:\#RPARS NIL)
              (IL:**COMMENT**FLG NIL))
             (DECLARE (GLOBAL IL:FILELINELENGTH IL:PRETTYFLG))
             (DECLARE (SPECIAL IL:FONTCHANGEFLG IL:\#RPARS IL:**COMMENT**FLG IL:*PRINT-SEMICOLON-COMMENTS*))
             (WITH-OPEN-FILE (STREAM (MAKE-PATHNAME :TYPE "LISP" :VERSION :NEWEST :DEFAULTS PATHNAME)
                                     :DIRECTION :OUTPUT)
                    (IL:LINELENGTH LINELENGTH STREAM)
                    (IL:RESETVARS
                     ;; Interlisp gorp that controls pretty printing
                     ((IL:FILELINELENGTH LINELENGTH)
                      (IL:PRETTYFLG T))
                     ;; First printout mode-line
```

```
                                (FORMAT STREAM "~A~%" MODE-LINE)
                    ;; Identifier
                                (FORMAT STREAM "~2%;;; File converted on ~A from source ~A" (IL:DATE)
                                       ROOT-NAME)
                                (LET ((DATES (GET ROOT-NAME 'IL:FILEDATES)))
                                     (WHEN DATES
                                         (FORMAT STREAM "~&~%;;; Original source ~A created ~A" (CDAR DATES)
                                                 (CAAR DATES))))
                                (TERPRI STREAM)
                                (TERPRI STREAM)
                    ;; Copyright notice
                                (LET ((OWNER (GET ROOT-NAME 'IL:COPYRIGHT)))
                                     (WHEN (AND OWNER (CONSP OWNER))
                                         (FORMAT STREAM "~&~%;;; Copyright (c) ")
                                         (DO ((TAIL (CDR OWNER)
                                                    (CDR TAIL)))
                                             ((NULL TAIL))
                                           (FORMAT STREAM "~4d" (CAR TAIL))
                                           (IF (CDR TAIL)
                                               (PRINC ", " STREAM)))
                                         (FORMAT STREAM " by ~a~%" (CAR OWNER))))
                                (TERPRI STREAM)
                    ;; Provide form
                                (PPRINT `(PROVIDE ,(STRING FILENAME))
                                       STREAM)
                                (TERPRI STREAM)
                    ;; In-package form
                                (AND PACKAGE-FORM (PPRINT PACKAGE-FORM STREAM))
                                (FORMAT STREAM "~2%;;; Shadow, Export, Require, Use-package, and Import forms
                                       should follow here~2%")
                                (DOLIST (COM (SYMBOL-VALUE (IL:FILECOMS ROOT-NAME)))
                                    (DOLIST (FORM (GET-COMS-FORMS COM STREAM))
                                        (PPRINT FORM STREAM)
                                        (TERPRI STREAM)
                                        (IL:BLOCK))))
                            (NAMESTRING STREAM)))))

(DEFUN CONSTRUCT-MODE-LINE (ROOT-NAME PACKAGE PACKAGE-P READTABLE READTABLE-P PRINT-BASE PRINT-BASE-P)
   (LET* ((DEFINE-FILE-ENVIRONMENT-FORM (LET ((NAME (CAR (IL:FILECOMSLST ROOT-NAME 'FILE-ENVIRONMENTS))))
                                          (AND NAME (REMOVE-COMMENTS (IL:GETDEF NAME 'FILE-ENVIRONMENTS
                                                                                'IL:CURRENT)))))
          (MAKEFILE-ENVIRONMENT (GET ROOT-NAME 'IL:MAKEFILE-ENVIRONMENT))
          (PACKAGE-FORM (SECOND (OR (MEMBER :PACKAGE DEFINE-FILE-ENVIRONMENT-FORM :TEST #'EQ)
                                    (MEMBER :PACKAGE MAKEFILE-ENVIRONMENT :TEST #'EQ))))
          (READTABLE-FORM (SECOND (OR (MEMBER :READTABLE DEFINE-FILE-ENVIRONMENT-FORM :TEST #'EQ)
                                      (MEMBER :READTABLE MAKEFILE-ENVIRONMENT :TEST #'EQ))))
          (BASE-FORM (SECOND (OR (MEMBER :BASE DEFINE-FILE-ENVIRONMENT-FORM :TEST #'EQ)
                                 (MEMBER :BASE MAKEFILE-ENVIRONMENT :TEST #'EQ))))
          SET-PACKAGE-FORM MODE-LINE-PACKAGE-FORM MODE-STRING)
         (WHEN (AND (NULL PACKAGE-P)
                    PACKAGE-FORM)
               (SETQ PACKAGE PACKAGE-FORM))
         (IF (PACKAGEP PACKAGE)
             (SETQ PACKAGE (PACKAGE-NAME PACKAGE)))
         (SETQ SET-PACKAGE-FORM (COND
                                  ((STRINGP PACKAGE)
                                   (SETQ MODE-LINE-PACKAGE-FORM PACKAGE)
                                   `(IN-PACKAGE ,PACKAGE))
                                  ((AND (CONSP PACKAGE)
                                        (EQ (CAR PACKAGE)
                                            'DEFPACKAGE))
                                   (LET ((NAME (STRING (SECOND PACKAGE)))
                                         (USE-LIST (CDR (ASSOC :USE PACKAGE :TEST #'EQ)))
                                         (NICKNAMES (CDR (ASSOC :NICKNAMES PACKAGE :TEST #'EQ)))
                                         (EXPORTS (CDR (ASSOC :EXPORT PACKAGE :TEST #'EQ)))
                                         FORM)
                                        (SETQ FORM
                                              `(IN-PACKAGE ,NAME
                                                      ,@(IF USE-LIST
                                                            `(:USE ',USE-LIST))
                                                      ,@(IF NICKNAMES
                                                            `(:NICKNAMES ',NICKNAMES))))
                                        (SETQ PACKAGE NAME)
                                        (SETQ MODE-LINE-PACKAGE-FORM
                                              `(,PACKAGE ,@(IF USE-LIST
                                                               `(":USE" ,USE-LIST))
                                                      ,@(IF NICKNAMES
                                                            `(":NICKNAMES" ,NICKNAMES))))
                                        (IF EXPORTS
                                            `(PROGN ,FORM (EXPORT ',EXPORTS))
                                            FORM)))
                                  ((AND (CONSP PACKAGE)
```

```
                                    (EQ (CAR PACKAGE)
                                        'IN-PACKAGE))
                                (LET ((NAME (STRING (SECOND PACKAGE)))
                                      (USE-LIST (EVAL (CADR (MEMBER :USE PACKAGE :TEST #'EQ))))
                                      (NICKNAMES (EVAL (CADR (MEMBER :NICKNAMES PACKAGE :TEST #'EQ))))
                                      FORM)
                                    (SETQ FORM PACKAGE)
                                    (SETQ PACKAGE NAME)
                                    (SETQ MODE-LINE-PACKAGE-FORM
                                          `(,PACKAGE ,@(IF USE-LIST
                                                           `(":USE" ,USE-LIST))
                                                     ,@(IF NICKNAMES
                                                           `(":NICKNAMES" ,NICKNAMES))))
                                    FORM))
                               (T (ERROR "Can't parse package form: ~s" PACKAGE))))
        (WHEN (AND (NULL READTABLE-P)
                   READTABLE-FORM)
              (SETQ READTABLE READTABLE-FORM))
        (IF (READTABLEP READTABLE)
            (SETQ READTABLE (IL:READTABLEPROP READTABLE 'IL:NAME)))
        (IF (STRING= READTABLE "XCL")
            (SETQ READTABLE "LISP"))
        (WHEN (AND (NULL PRINT-BASE-P)
                   BASE-FORM)
              (SETQ PRINT-BASE BASE-FORM))
        (IF (NOT (TYPEP PRINT-BASE '(INTEGER 0 *)))
            (ERROR "Incorrect print-base form: ~s" PRINT-BASE))
        (SETQ MODE-STRING (CONCATENATE 'STRING ";;;-*- Package: " (PRINC-TO-STRING MODE-LINE-PACKAGE-FORM)
                                  "; Syntax: "
                                  (IF (STRING= READTABLE "LISP")
                                      "Common-Lisp"
                                      READTABLE)
                                  "; Mode: Lisp; Base: "
                                  (PRINC-TO-STRING PRINT-BASE)
                                  " -*-"))
        (VALUES PACKAGE READTABLE PRINT-BASE MODE-STRING SET-PACKAGE-FORM)))

(DEFUN GET-COMS-FORMS (COMMAND STREAM)                    ; Edited 2-Aug-88 15:37 by ht:
   (LET
    ((UNSUPPORTED-TYPES '(IL:FNS IL:SPECVARS IL:GLOBALVARS IL:LOCALVARS IL:INITVARS IL:ALISTS IL:DEFS
                                 IL:INITRECORDS IL:LISPXMACROS IL:MACROS IL:PROPS IL:RECORDS IL:SYSRECORDS
                                 IL:USERMACROS IL:VARS IL:CONSTANTS EXPORT IL:RESOURCES IL:INITRESOURCES
                                 IL:GLOBALRESOURCES IL:I.S.OPRS IL:HORRIBLEVARS IL:UGLYVARS IL:BITMAPS IL:CURSORS
                                 IL:ADVICE IL:ADVISE IL:COURIERPROGRAMS IL:TEMPLATES))
     (FILEPKGTYPE (CAR COMMAND)))
    (IF (MEMBER FILEPKGTYPE UNSUPPORTED-TYPES :TEST #'EQ)
        (LIST (MAKE-COMMENT "Filepkg type ~s not supported: ~s" FILEPKGTYPE COMMAND))
        (CASE FILEPKGTYPE
            (IL:P (CDR COMMAND))
            (IL:E
               ;; done this way so the comment doesn't get in the way of any tricky printing done under the E
               (PPRINT (MAKE-SEMICOLON-COMMENT (FORMAT NIL "~S" COMMAND)
                               1)
                       STREAM)
               (LET ((*STANDARD-OUTPUT* STREAM))
                    (MAPC #'EVAL (CDR COMMAND)))
               NIL)
            (IL:COMS
               ;; Recurse
               (MAPCAN #'(LAMBDA (X)
                                 (GET-COMS-FORMS X STREAM))
                       (CDR COMMAND)))
            ((EVAL-WHEN IL:EVAL-WHEN) `((EVAL-WHEN ,(MAPCAR #'(LAMBDA (SYM)
                                                                 (INTERN (STRING SYM)
                                                                         (FIND-PACKAGE "LISP")))
                                                            (SECOND COMMAND))
                                                   ,@(MAPCAN #'(LAMBDA (X)
                                                                       (GET-COMS-FORMS X STREAM))
                                                             (CDDR COMMAND)))))
            (IL:DECLARE\: (WITH-COLLECTION (LET ((CONTEXT '(LOAD EVAL)))
                                             (DOLIST (TOKEN (CDR COMMAND))
                                                 (CASE TOKEN
                                                     ((IL:COPY IL:DOCOPY) (PUSHNEW 'LOAD CONTEXT))
                                                     ((IL:DOEVAL@COMPILE IL:EVAL@COMPILE)
                                                        (PUSHNEW 'COMPILE CONTEXT))
                                                     ((IL:DOEVAL@LOAD IL:EVAL@LOAD) (PUSHNEW 'EVAL CONTEXT))
                                                     ((IL:DONTCOPY) (SETQ CONTEXT (REMOVE 'LOAD CONTEXT)))
                                                     ((IL:DONTEVAL@COMPILE) (SETQ CONTEXT (REMOVE
                                                                                           'COMPILE CONTEXT))
)
                                                     ((IL:DONTEVAL@LOAD) (SETQ CONTEXT (REMOVE 'EVAL CONTEXT)
                                                                               ))
                                                     ((IL:FIRST IL:NOTFIRST IL:EVAL@LOADWHEN
                                                            IL:EVAL@COMPILEWHEN IL:COPYWHEN IL:COMPILERVARS)
```

```
;; IGNORE
 (WARN "Ignoring ~s declaration" TOKEN))
                                                    (OTHERWISE (COLLECT `(EVAL-WHEN ,CONTEXT
                                                                          ,@(GET-COMS-FORMS TOKEN
                                                                                            STREAM)))))))))
            ((IL:*)
               ;; Comment
               (LIST COMMAND))
            (IL:FILES (LET ((FILE-NAMES (MAPCAN #'(LAMBDA (TOKEN)
                                                    (IF (NOT (CONSP TOKEN))
                                                        (LIST TOKEN)))
                                                (REMOVE-COMMENTS (CDR COMMAND)))))
                        `(,(MAKE-COMMENT "Translated ~s to require forms" COMMAND)
                          ,@(WITH-COLLECTION (DOLIST (FILE FILE-NAMES)
                                                     (COLLECT `(REQUIRE ,(STRING FILE))))))))
            (IL:PROP
               ;; Throw out makefile props
               (LET ((PROPS (SECOND (REMOVE-COMMENTS COMMAND))))
                    (IF (NOT (LISTP PROPS))
                        (SETQ PROPS (LIST PROPS)))
                    (IF (SET-DIFFERENCE PROPS '(IL:FILETYPE IL:MAKEFILE-ENVIRONMENT))
                        (MAKE-COMMENT "Ignoring prop ~s coms: ~s" PROPS COMMAND))))
            (T ;; Should the filepkgtype of a definer
               (LET
                ((IGNORED-DEFINERS '(FILE-ENVIRONMENTS IL:DEFINE-TYPES OPTIMIZERS IL:SEDIT-FORMATS
                                     ADVISED-FUNCTIONS IL:COMMANDS IL:SPECIAL-FORMS PROFILES
                                     WALKER-TEMPLATES))
                 (DEFINER-TYPE (IL:GETFILEPKGTYPE FILEPKGTYPE 'IL:COMMANDS T)))
                (IF (MEMBER DEFINER-TYPE IGNORED-DEFINERS :TEST #'EQ)
                    (UNLESS (EQ DEFINER-TYPE 'FILE-ENVIRONMENTS)
                        (LIST (MAKE-COMMENT "Ignoring definer coms: ~s" COMMAND)))
                    (LET*
                     ((GET-DEF-METHOD (AND DEFINER-TYPE (GET DEFINER-TYPE :DEFINED-BY)
                                           (GET DEFINER-TYPE 'IL:GETDEF)))
                      (DEFS (AND GET-DEF-METHOD (MAPCAR #'(LAMBDA (NAME)
                                                            (IF (COMMENT-P NAME)
                                                                NAME
                                                                (FUNCALL GET-DEF-METHOD NAME DEFINER-TYPE))
                                                            )
                                                        (CDR COMMAND)))))
                      (SETQ
                       DEFS
                       (CASE DEFINER-TYPE
                           (IL:FUNCTIONS
                              ;; Transform defdefiners to defmacros
                              (MAPCAN #'(LAMBDA (DEF)
                                          (IF (AND (NOT (COMMENT-P DEF))
                                                   (EQ (CAR DEF)
                                                       'DEFDEFINER))
                                              (LET* ((CLEANED-FORM (REMOVE-COMMENTS DEF))
                                                     (NAME (SECOND CLEANED-FORM))
                                                     (DEFINER-FOR (THIRD CLEANED-FORM))
                                                     (BODY (CDR (MEMBER DEFINER-FOR DEF))))
                                                    (LIST (MAKE-COMMENT "Transforming defdefiner (~s ~s ~s
                                                                        ... ) to defmacro" (FIRST DEF)
                                                                        (SECOND DEF)
                                                                        (THIRD DEF))
                                                          `(DEFMACRO ,(IF (CONSP NAME)
                                                                          (CAR NAME)
                                                                          NAME) ,@BODY)))
                                              (LIST DEF)))
                                      DEFS))
                           (OTHERWISE DEFS)))
                      (OR DEFS (LIST (MAKE-COMMENT "Can't parse: ~s" COMMAND)))))))))))


(DEFUN MAKE-COMMENT (&REST ARGS)
   (APPLY #'WARN ARGS)
   (MAKE-SEMICOLON-COMMENT (APPLY #'FORMAT NIL ARGS)
          1))


(DEFINE-FILE-ENVIRONMENT "XCL-BRIDGE" :PACKAGE "XCL"
   :READTABLE "XCL"
   :COMPILER :COMPILE-FILE)

;; comment identity preservation hack

(DEFPARAMETER *PRESERVE-COMMENT-START-CHAR* #\.)
```

```
(DEFPARAMETER *PRESERVE-COMMENT-START-CHARCODE* 46
   "used at beginning of comments to preserve comment start info if IL:*PRINT-SEMICOLON-COMMENTS* is :PRESERVE")


(DEFUN INITIAL-COMMENT-LINE-P (STRING)                          ; Edited 10-Aug-88 10:17 by ht:
   (AND (> (LENGTH STRING)
           0)
        (EQ (CHAR STRING 0)
            *PRESERVE-COMMENT-START-CHAR*)))


(DEFUN FIX-COMMENT-? (X)                                        ; Edited 10-Aug-88 10:23 by ht:
   (WHEN (AND (COMMENT-P X)
              (INITIAL-COMMENT-LINE-P (CADDR X)))
      ;; remove the preserve key char and the following spaces, if any
      (IL:GNC (CADDR X))
      (LOOP (IF (EQ (IL:NTHCHARCODE (CADDR X)
                                    1)
                    32)
                (IL:GNC (CADDR X))
                (RETURN)))))

(REINSTALL-ADVICE '(IL:CONCAT :IN IL:PRIN2-LONG-STRING)
      :AFTER
      '((:LAST (COND
                  ((EQ IL:*PRINT-SEMICOLON-COMMENTS* ':PRESERVE)
                   (IL:RPLCHARCODE IL:!VALUE -1 *PRESERVE-COMMENT-START-CHARCODE*))))))

(REINSTALL-ADVICE '(IL:PRIN1 :IN IL:PRIN2-LONG-STRING)
      :AFTER
      '((:LAST (COND
                  ((EQ IL:X IL:SEMISTRING)
                   (IL:RPLCHARCODE IL:X -1 32))))))

(IL:PUTPROPS XCL-BRIDGE IL:COPYRIGHT ("Xerox Corporation" 1988 1989))
```

FUNCTION INDEX

VARIABLE INDEX

ADVICE INDEX

FILE-ENVIRONMENT INDEX